COMPUTER BASED BEHAVIORAL BIOMETRIC AUTHENTICATION

VIA MULTI-MODAL FUSION

THESIS

Kyle O. Bailey, Second Lieutenant, USAF

AFIT-ENG-13-M-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-13-M-04

COMPUTER BASED BEHAVIORAL BIOMETRIC AUTHENTICATION VIA MULTI-MODAL FUSION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyberspace Operations

Kyle O. Bailey, B.S.C.S.

Second Lieutenant, USAF

March 2013

AFIT-ENG-13-M-04

COMPUTER BASED BEHAVIORAL BIOMETRIC AUTHENTICATION

VIA MULTI-MODAL FUSION

Kyle O. Bailey, B.S.C.S.
Second Lieutenant, USAF

Approved:

Gilbert L. Peterson, PhD (Chairman)	27 FEB 2013 Date
Jeffrey D. Clark, Lieutenant Colonel, PhD (Member)	27 Feb 2013 Date
Kennard R. Laviers, Major, PhD (Member)	1 Mar 2013 Date

AFIT-ENG-13-M-04

Abstract

Biometric computer authentication has an advantage over password and access card authentication in that it is based on something you *are*, which is not easily copied or stolen. One way of performing biometric computer authentication is to use behavioral tendencies associated with how a user interacts with the computer. However, behavioral biometric authentication accuracy rates are much larger then more traditional authentication methods. This thesis presents a behavioral biometric system that fuses user data from keyboard, mouse, and Graphical User Interface (GUI) interactions. Combining the modalities results in a more accurate authentication decision based on a broader view of the user's computer activity while requiring less user interaction to train the system than previous work. Testing over 30 users, shows that fusion techniques significantly improve behavioral biometric authentication accuracy over single modalities on their own. Two fusion techniques are presented, feature fusion and decision level fusion. Using an ensemble based classification method the decision level fusion technique improves the False Acceptance Rate (FAR) by 0.86% and False Rejection Rate (FRR) by 2.98% over the best individual modality.

Acknowledgments

I would like to thank James Okolica for creating the Windows driver used for data collection, anyone who contributed their time to participating as a test subject, and 1^{st} Lieutenant Alanna Keith for being the testing proctor.

Kyle O. Bailey

Table of Contents

Page

List of Figures

List of Tables

List of Symbols

Symbol	Definition
γ	gamma parameter used by LibSVM

List of Acronyms

Acronym	Definition
AFIT	Air Force Institute of Technology
HTML	HyperText Markup Language
FAR	False Acceptance Rate
FRR	False Rejection Rate
EER	Equal Error Rate
GUI	Graphical User Interface
ROC	Receiver Operating Characteristic
CAC	Common Access Card
SVM	Support Vector Machine
ANN	Artificial Neural Network
SFS	Sequential Feature Selection
PMMR	Plus-M-Minus-R
RBF	Radial Basis Function
IDS	Intrusion Detection System
API	Application Programming Interface
ANOVA	Analysis of Variance
EBDL	Ensemble Based, Decision Level

COMPUTER BASED BEHAVIORAL BIOMETRIC AUTHENTICATION
VIA MULTI-MODAL FUSION

I. Introduction

COMPUTER systems have become increasingly integral to the way that information is created and transferred in our society, making the security of these systems more important than ever. Information security is commonly separated into three categories, confidentiality, integrity and availability, each of which are equally important when considering the security of data [3]. The authenticity of a user who is accessing that data must be validated in order for all three information security principles to hold.

Authentication of a user is the process of confirming that the individual accessing and interacting with the computer, is who they claim to be. Traditionally authentication is based on something you know and/or something you have. An example would be using a Common Access Card (CAC) and pin number or a username and password [34]. One downside however is that this type of authentication can be lost, stolen, or disclosed. It also does not truly identify the user as themselves, but instead by something they know or have. Biometric authentication is advantageous in that it is based on something you *are* [9]. There are two subsets of biometric authentication, physiological and behavioral [25]. These authentication methods identify the user as themselves based on measurable physical or behavioral characteristics.

1.1 Physiological Biometrics

Physiological biometric authentication involves measuring physical characteristics of a persons body that make them unique. Physiological methods include fingerprint scanning, facial recognition, hand geometry recognition or retinal scans [25]. Generally these methods are more reliable and successful in real world application than behavioral techniques [14]. One drawback of physical biometrics is that they require hardware to perform the biometric data collection. This hardware adds cost and another layer of complexity to the login process for the user. Another drawback is that all of the physical biometric methods still contain some error. Comparison testing by Bhattacharyya, et al. [25], found that the iris scanner, with an Equal Error Rate (EER) of 0.01% performed the best.

1.2 Behavioral Biometrics

Behavioral biometric authentication is the process of measuring behavioral tendencies of a user resulting from both psychological and physiological differences from person to person. Behavioral methods for authentication include typing dynamics [1, 2, 6, 10, 20, 37], mouse dynamics [11, 12, 14, 18, 19, 39], voice recognition [25], signature verification [25] and Graphical User Interface (GUI) usage analysis [7, 9, 15–17, 22]. Due to the variability of the human body and mind, the adoption of this type of biometrics has lagged behind physiological biometrics. However, the use of keystrokes, mouse dynamics and GUI interaction for biometrics does not require extra hardware. The data can be collected using software that gathers information from the existing keyboard, mouse and GUI messages sent by the installed operating system. A second benefit to computer interaction based biometrics is that authentication can occur actively throughout the user's session as opposed to once during initial logon. This could prevent a user's session from being hijacked after the initial logon has occurred.

1.2.1 Issues with Behavioral Biometrics.

Current implementations of behavioral biometric authentication systems have underlying problems resulting in slow adoption into real world environments. The first is the amount of user data that is needed to both train and test the system. Previous systems need thousands of user actions which can take hours of interaction before a decision is able to be made to the claimed accuracy. Keystroke based systems need to be trained on 15,000-85,000+ keystrokes and tested on 300-900 keystrokes. Mouse movement based systems need 10,000-12,500 mouse movements for their training set and 25-2,000 movements for testing. GUI thresholds have yet to be established by previous work. This amount of data can take time to collect and might not catch a malicious user who is trying to minimize their time on the system.

Second, is the number of modalities that previous systems collect data from. Almost all previous systems collect data from only one modality leaving others susceptible to malicious use without the biometric system knowing.

Finally the accuracy of the system needs to be improved. High number of false alarms frustrates users and network administrators but on the other hand a lack of detection of malicious use cannot be tolerated. Both types of error need improvement from current levels and should begin to approach physical biometric error rates if behavioral biometrics are ever to be adopted.

1.3 Multi-Modal Fusion

This thesis presents a behavioral biometric system that fuses user data from keyboard, mouse, and GUI interactions. The system collects user characteristics relating to the way a particular user interacts with the computer. This is done by monitoring a users keystrokes, mouse movements, and GUI usage patterns while they are performing free computer use over a set of three research based tasks. Features are calculated on these actions. Feature fusion is then used to combine data from the three modalities for classification.

Classification occurs on the data in both an identification (multi-class) and authentication (binary class) situation leading to the reported results. Identification is the process of determining who the user is, while authentication is used to confirm the validity of that identity. Additionally, an Ensemble Based, Decision Level (EBDL) fusion method is analyzed which first classifies on each modality alone and generates a fusion of the results. A final classifier then produces a decision. From the experiments that follow, it was found that using EBDL fusion, significant classification improvements were achieved over each of the individual modalities on their own and feature fusion.

By collecting user interactions from all three surfaces, the malicious user cannot escape the watchful eye of a system that is able to monitor all at once. For example, if a malicious user knows that keystroke dynamics are being used to monitor a computer system, they could perform their activities by only touching the mouse. On top of this GUI usage analysis seeks to emphasize how the user interacts with the system, such as do they prefer keyboard shortcuts over GUI menus, page up/down versus the scroll bar or scroll wheel, etc. There are thousands of minute differences between how two different users interact with a computer system. Analyzing the entire picture of a users interaction is shown to improve the accuracy and reliability of a behavioral biometric system.

1.4 Thesis Structure

This thesis is laid out in the following structure. Chapter 2 presents prior work on biometrics, with the focus being on keystrokes, mouse dynamics, and GUI usage analysis. Previous work that combines multiple biometric techniques are also discussed along with concerns associated with using biometrics as an authentication method. Chapter 3 discusses the experimental design of the fusion system to include the data collection method, participant selection and tasks, features generated from the data, and two methods of fusing data from multiple modalities. Chapter 4 discusses the performance achieved by the individual modalities and both fusion methods, along with a comparison of the fusion system against previous work, and an analysis of how demographics may affect the fusion technique. Chapter 5 includes future work and final thoughts.

II. Related Work

The idea of using keystroke and mouse dynamics as a supplement to traditional authentication has been around for several decades [31], but there has been minimal research done in the area of combining these two techniques into one system. Graphical User Interface (GUI) usage analysis is a relatively young technique [16] and brings in the concept of trying to analyze exactly how the user accomplishes a task within the operating system interface.

The chapter presents the way biometrics are measured, a definition of identification and authentication, and static versus dynamic authentication. This is followed by previous work done in the use of keystroke dynamics, mouse dynamics, and GUI usage analysis as a means for authentication. Previous work that has fused multiple biometric modalities is also discussed. Finally concerns with the use of biometrics as an authentication method are explored.

2.1 Metrics for Biometric Authentication

Being able to quantify the effectiveness of the authentication technique is important. Previously, performance has been measured using the metrics of False Acceptance Rate (FAR), False Rejection Rate (FRR) and the Equal Error Rate (EER). Both FAR and FRR are reported as a percentage, and signify the percentage of time an impostor is authenticated (FAR) or the percentage of time a legitimate user is denied access (FRR). The EER is the value where at the FAR and FRR are equal. This point is determined by creating a curve for both FAR and FRR based on the Receiver Operating Characteristic (ROC) for the classification algorithm [25]. An example of some of the ROC's used are the number of seconds of interaction [11], the number of mouse events [12], and the number of keystrokes [20]. A majority of the previous work reports their results in either FAR and FRR or EER but there are a few [2, 22] who use the rate of identification or detection as a metric for performance. These values are simply recorded as the percentage of time the system can make a correct decision on the identity or authentication status of the user. It is important to note that these values can heavily depend on the number of tests performed.

2.2 Identification and Authentication

Previous work has defined two different problems to be solved by biometrics, identification and authentication. These two problems have been defined several ways but in this thesis they will be discussed in the following context. Identification is the process of determining the identity of a user [9]. This means the system will come up with an answer of who provided the data sample based on its stored database of known users. Authentication on the other hand, confirms whether or not that identity is valid [9]. In biometrics, when performing authentication the system will either give a yes or no answer based on whether the sample provided matches it's known user. Traditional authentication systems such as a username and password, perform both of these functions, with the username acting as a form of identification and the password being used to confirm this identity.

2.3 Static vs. Dynamic Authentication

With the focus being on authentication, it is necessary to note the two different ways that authentication can be performed, statically or dynamically. Static authentication is what most computer users are familiar with. During static authentication user verification is performed only once, when the user enters their password at logon or into a lock screen. This subsequently leaves the session open for attack, and there is little way of determining who has control of the keyboard. Dynamic or active authentication on the other hand, is ongoing throughout the users session and would ideally prevent an unauthorized user from taking control of the computer once the static authentication phase has been completed.

There have been several different methods that use behavioral biometric techniques to perform dynamic authentication. These include monitoring the users keystroke dynamics [1, 2, 6, 10, 20, 37], mouse dynamics [11, 12, 14, 19] or GUI interaction style [7, 9, 15–17, 22] just to name a few of interest.

In order to simulate dynamic authentication in a testing environment, previous research has elected to divide up a users session using varying techniques. Ahmed, et al. [13, 14] and Zheng, et al. [12] both divided up a users session based on the number of actions they wanted per *block* whereas Marsters, et al. [20] and Garg, et al. [22] divided up the users session on a set time interval with a minimum number of actions required in that time interval. Finally Imsand [9, 15–17] and Pusara, et al. [7] cut up the users data into thirds or quarters respectively using part for classification and leaving one fraction out for verification testing. This is an important variable to note for each previous work in order to understand how they were seeking to achieve dynamic authentication.

2.4 Keyboard Dynamics

Gaines, et al. [31], introduced the idea of using behavioral biometrics as a supplement to traditional authentication. Initially, keystroke timing data was used to supplement password entry [10, 31, 37], this evolved into being able to analyze long structured text as a basis for authentication [1, 2], and finally long free text samples [5, 6, 20]. Each work mentioned used a similar set of features for classification which include intra-key timing, or the latency between the depress of one key to the next, and key hold duration, or the average time between when a key is depressed and released. Research has been done using several statistical classifiers which have attained similar results in terms of classification accuracy [20]. An overview of the features that each paper calculated and the type of classifier they used can be seen in Table 2.1. All work discussed below used a standard 104 key keyboard with their participants typing in English unless otherwise noted.

Joyce, et al. [10] proposed a method for using keystroke latencies in order to create a unique digital signature. This signature was created by requiring the user type in their first and last name as well as their username and password eight separate times during an enrollment phase. This established a baseline of latencies between when the different key pairs were pressed. For example, if the username was Jim there would be timing information between the press of the J and I, and I and M. From this a threshold is established which is set as two standard deviations outside of the mean calculated from the enrollment values for each latency. If any of the latency values fall outside of this range the person is rejected access and they are deemed to be an impostor. Since this work was solely password based it does not mention dynamic authentication but rather focuses on enhancing the static authentication portion of a logon. Using this method Joyce, et al. [10], were able to achieve a FAR of 16.25% and a FRR of 0.25% in their study.

Brown, et al. [37] similarly focused on using short strings to collect typing dynamics, specifically the users name, but used a neural network for the classifier as opposed to the statistical method used by Joyce, et al [10]. In the experiment 46 test subjects were asked to type their name 25 times and then also type the names of the other subjects in order to provide imposter data. This timing information was used to create a reference model to train the classifier. Like Joyce, et al. [10] the features calculated for each entry included the latencies between the individual key presses and any samples that were deemed as outliers relative to that users data set were thrown out. Two different Artificial Neural Network (ANN)'s were tested, the Adaptive Linear Element (ADALINE) and a backpropagation neural network. The best results were achieved using a backpropagation neural network that was partially connected and trained to produce the lowest possible FAR. Using this setup and technique Brown, et al. [37], were able to achieve a FAR of 0.0% and a FRR of 12.0%.

This was followed up by Monrose, et al. [1, 2]. Their work on using keystroke dynamics as a biometric for authentication showed promising results based on some modifications of the work done by Joyce et al. [10]. They received structured typing samples (100-200 words) from 63 different test subjects over a period of 11 months. The latency between key presses and the duration of each press was recorded for all test subjects and stored as profiles. Latencies of the most common key pairs and key triads were then used to calculate features for the classifier. For example some of the features could be timing information between th, er, in, on, ng, are, ing, etc. Three different types of classifiers were then tested but the weighted probabilistic classifier showed the best results. This type of classifier uses probabilities based on the number of times a given feature was seen and added more weight to digraph features such as th, er, in, etc., that had higher frequencies in the English language. A reference score is then calculated between each reference profile and the unknown profile. The profile that generates the largest reference score is then labeled to be the same user as the unknown profile. Monrose et al. [1, 2], were able to report an identification success rate of 87.18% over all 63 test subjects.

Gunetti, et al. [6], proposed the first system that used only free text typing samples. The collection of data by Gunetti, et al. [6], was done through a HyperText Markup Language (HTML) based web page that used Javascript to capture the timing data. Test subjects were allowed to enter anything they desired into the web page but were asked to type anywhere from 700 to 900 characters. It should be noted that all test subjects spoke Italian and were asked to type their samples in Italian. This research did not focus on dynamic authentication and therefore calculated the distance measures on each individual sample of text as a whole. Forty test subjects provided 15 samples each, over the course of 11 months. Two methods of calculating the distance between two typing samples were developed by Gunetti, et al. [6] as well as Bergadano, et al. [5], called *R* measures and *A* measures. *R* measures are calculated by first determining the digraphs two typing samples have in common, ranking these in descending order for each user based on latency time and then determining the normalized disorder of one of the samples when considering the other sample as ordered. A threshold was then used to determine at what disorder value two samples should be considered different. *A* measures are calculated by taking the latency times of digraphs that two samples have in common, the larger of the two latency times are divided by the smaller latency time. If this ratio is larger than a threshold that is set than the two samples are dissimilar for that digraph. These distance measures were used in different combinations between digraphs, trigraphs and four-graphs as a classifier. Using the *R* measure on digraphs, trigraphs and four-graphs as well as the *A* measure on digraphs Gunetti, et al. [6], were able to achieve a FAR of 0.005% and a FRR of 5.0%.

Finally, Marsters [20], developed a system called BAKER (Biometric Analysis of Keystroke Entry Rhythms) which used a Bayesian network classifier to provide active authentication using keystroke dynamics. Data was collected from 10 test subjects who installed the software on their machine and left it running for as long as 18 months. Marsters initial plan was to calculate both duration of the key hold and the latency between key presses but unlike previous research the surrounding context was also taken into account. For example to determine the duration of a key hold the letter pressed before and the letter pressed after were also recorded such that the hold time for "key Y was *n* milliseconds when preceded by key X and succeeded by key Z" [20]. This results in a trigraph of data. A quadgraph is collected for the latency between key presses but takes on the form "the latency between the press of key X and key Y was *n* milliseconds when preceded by key W and succeeded by key Z". Due to a limitation of their hardware however, Marsters had to eliminate the contextual information and proceed with unigraphs for duration and digraphs for latency information. The mean value recorded was stored for each type of feature. Several different classifiers were tested but the best results, both in terms of speed and classification ability, were seen from a BayesNet classifier which is integrated into the Weka data mining toolkit. For their system to perform active authentication a three hour time slice was set with a requirement that each three hour block contain at least 300 keystrokes. Three hours is one of the longest time slices seen in previous work and was presumably picked to allow for the collection of ample data to calculate meaningful unigraphs and digraphs but this is not explicitly stated. From this research an EER of 0.27% was reported.

Even though there has been a fair amount of research into keystroke dynamics it has been suggested that there remains much to be desired in terms of implementation and deployment on a commercial level [20]. Several companies have released software that was designed to not only verify the users password but also the way the password was typed. This includes Deepnet Security's TypeSense, and bioChec which contains a patented signature matching algorithm [9].

Table 2.1: Keystroke features from previous work.

Article	**Features Calculated**	**Classifier**
Joyce, et al. [10]	Latencies between key presses of: - Username, password and full name	Custom built Statistical classifier
Brown, et al. [37]	Latencies between key presses of: - Full name	Neural Network
Monrose, et al. [2]	Duration and latency between: - 100 to 200 words of free or structured text	Weighted probabilistic classifier
Gunetti, et al. [6]	Duration and latency between: - 700 to 900 characters of free text	*R* and *A* distance measures
Marsters [20]	Duration and latency between: > 300 characters of free text	BayesNet

2.5 Mouse Dynamics

Biometrics based on mouse dynamics involves monitoring the way a user moves the mouse in order to use that data as a means for authentication [11, 12, 14, 19]. Initially Gamboa, et al. [11], used a memory game to capture mouse movements, where as work that came after ([12, 14, 19]) focused on free mouse movements in day-to-day use. The features calculated on this type of data include average speed per movement direction, click based interval times, action histogram, and average movement speed per travel distance. Classification was performed using both an ANN [14] and Support Vector Machine (SVM) [12, 19].

Gamboa, et al. [11], proposed a behavioral biometric system based on human interaction with the pointing device. They constructed a web page and corresponding script that collected a user's mouse coordinates as they were playing a memory game. Due to this they were not able to make a guarantee on the type of mouse their participants were using whether it be three button mouse, track pad, etc. Gamboa, et al. collected about 10 hours of interaction from 50 users, which corresponded to about 400 individual mouse strokes per user. They defined a stroke as the movement between two points in their memory game. They then calculated 63 features for each stroke from this data, which can be seen in Table 2.2. This was done for 50 strokes with the average of all the values taken to make one *block* of features. The most discriminating features were selected using a greedy Sequential Feature Selection (SFS) method and then fed into a sequential classifier. To train the classifier they calculated features over the first half of the data collected and the second half was used for testing the performance of the system. By using this method, Gamboa, et al. [11] were able to achieve an EER of 2% using ninety seconds of interaction.

Ahmed, et al. [14], used real operating conditions and monitored a user's mouse activity during their daily activity. The data collection was performed by installing interception software on each of the user's workstations, which recorded mouse activity every quarter of a second and periodically sent the data back to a detection server placed on their network. Due to the fact that participants used their own workstations, the type of mouse, pointer speed and screen resolution could not be controlled, however they do not think this affected their results. The interception software was installed on the machines of 22 participants leading to 998 sessions being recorded with an average of almost 13 hours of input per user. From this data Ahmed, et al. [14], generated 39 different features for each user, to include things like movement speed compared to direction, an action type histogram, movement type histogram and many more. The full list is displayed in Table 2.2. These features were then fed into an ANN which was used as the classifier. The neural network was configured to automatically place a higher weight to features that it deemed the most reliable or discriminating based on the input data. The neural network used was a feed-forward multilayer perception network consisting of three layers, and 39 nodes, trained using the Levenberg-Marquardt back propagation algorithm. In order to facilitate dynamic authentication in their experiment, the user's data was sliced into blocks of 2,000 actions. From their research, Ahmed, et al. [14], were able to achieve a FAR of 2.4649% and a FRR of 2.4614%.

Shen, et al. [19], also used mouse based biometrics with an SVM to try and further improve on the results seen by Ahmed et al. [14]. Shen, et al.[19], similarly asked participants to install a data collector on their workstation which would record their mouse activity during a normal day of computer use. Once again this implies that the exact type of mouse used by the participants was not controlled however it was presumed to remain constant over the testing period. They were able to collect data from 20 users over a period of two months recording at a frequency of 100 hertz. From this Shen et al. [19], calculated 45 features similar to those mentioned above and are displayed Table 2.2. They then used SFS and Plus-M-Minus-R (PMMR) to determine the most discriminating set of features for the data being used. The hypo-optimum features were tested on both an ANN and SVM.

Table 2.2: Mouse modality features from previous work.

Article	Features Calculated	Classifier
Gamboa, et al. [11]	Min, max, std dev, and min-max of each per stroke: - Horizontal and vertical coordinate vector - Angle of the path tangent with the x axis - Curvature - Horizontal, vertical, tangential and angular velocity - Tangential acceleration and jerk - Time and length of the stoke - Straightness, jitter, paused time and paused time ratio - High curvature critical points, and time to click	Sequential classifier
Ahmed, et al. [14]	- Movement speed compared to travel direction - Average movement speed per direction - Movement direction histogram - Average movement speed per action type - Travel distance histogram - Movement elapsed time histogram	ANN
Shen, et al. [19]	- Mouse action histogram - Mouse silence ratio - Distribution of actions on the screen - Distribution of movement distances - Distribution of movement directions - Single-click interval times - Double-click interval times - Average movement speed compared to travel distance - Extreme movement speed compared to travel distance	SVM
Zheng, et al. [12]	Angle metrics calculated between the endpoints of movements: - Direction: angle between the horizontal and line of travel - Angle of curvature (AOC): angle formed by three consecutive points A,B,C the AOC is angle ABC - Curvature distance: ratio of the distance from point A to C and perpendicular distance from B to the line AC - Movement speed - Pause and click time	SVM

Features were calculated on 30 evenly sized blocks for the *known* user and 45 total samples from nine different malicious users. By doing this Shen, et al. [19], were able to report a FAR of 1.86% and a FRR of 3.46% when using a SVM with the PMMR method of feature selection. Shen, et al. [18, 35, 39] also published three other articles on mouse dynamics which were not included in this thesis due to a lack of improvement on the results published in the work mentioned in this paragraph [19].

Zheng, et al. [12] derived a different approach that involved using a point by point calculation of angles from the endpoints of consecutive mouse movements. Two tests were performed, one that was controlled and a second that used a web interface and forum to capture the user's mouse movements meaning screen resolution, and mouse type could not be monitored. The controlled data set contained individuals who were asked to perform routine tasks on their workstation whereas the field group contained mouse movements from over 1,000 unique users on a forum page. From this data Zheng generated features on *blocks* of 25 point and click actions for a given user. The statistical features were calculated between each set of individual points and are listed in Table 2.2. To perform classification a SVM was used from the LibSVM 3.0 package with the Radial Basis Function (RBF) as the kernel function. Zheng, et al. were focused on dynamic authentication and made the point that their methods needed less user input data than any previous methods. They were able to achieve an EER of 1.3% when using a block size that contained 25 point and click actions however, it should be noted that to achieve these results 500 training blocks of 25 point and click actions were still needed per user.

2.6 Graphical User Interface Interaction

The core concept behind using a user's GUI interaction style for biometrics has its roots in a method called command line profiling. Command line profiling [40], monitors the commands a user sent a command line based system, such as UNIX, in order to create an Intrusion Detection System (IDS). The idea behind the concept was that different people use different sets of commands to perform the same core task. This work was followed up by Maxion, et al. [38], and Coull, et al. [8], who achieved promising results but were not accurate enough to create a deployable system. GUI interaction based biometrics can be thought of in the same manner as command line profiling. When a user wants to accomplish a task on the system, there are often many different modalities that can be used. This includes entirely different programs that perform the same end task, using keyboard shortcuts versus GUI buttons, etc. When thinking of interacting with the GUI by sending *commands* one can draw parallels between command line profiling and GUI interaction in terms of their use as a biometric technique. Previous work has shown that analyzing the modalities of a user's interaction result in a promising biometric technique [7, 9, 15–17, 22].

Pusara, et al. [7], did an initial look into using GUI based interaction. Their goal was to derive a method that could be used for continuous authentication or user re-authentication as Pusara called it. Mouse data was collected from 18 users while interacting with Internet Explorer. A total of 10,000 unique cursor locations were sampled which took about two hours per subject. The data collected included mouse wheel movements, single or double clicks and mouse movements outside of the Internet Explorer window. Similar features were calculated as in the work of Gamboa, et al. [11], but there was more of a focus on the number of times a specific action had occurred as opposed to how an individual action occurred. The full list of features calculated by Pusara, et al. [7], can be seen in Table 2.3. A C5.0 decision tree was used for classification without boosting through the data mining tool See5. User re-authentication was simulated by dividing up the user's session into quarters. The first two were used for training the classifier, the third for parameter selection and the last for testing. From their research Pusara, et al. [7], were able to achieve a FAR of 0.43% and a FRR of 1.75%.

Garg, et al. [22], expanded on the work of Pusara, et al. [7], but focused more on preventing masquerade attacks as opposed to user re-authentication. A masquerade attack is defined by Garg, et al. as an attack that occurs when a person exploits a user's credentials to access a system when they are not entitled to do so. Both user re-authentication [7], and masquerade attack have the same fundamental meaning however. Garg, et al. [22]

collected data from three users who varied from 9 to 50 collection sessions. The collection software that was used extracted information such as keyboard activity, mouse movements and events, as well as operating system related information such as when a process was created or terminated. From this information the features calculated were very mouse centric and can be seen in Table 2.3. There is however, no explanation as to why keyboard and operating system based features were not derived from the collected data. Garg used a SVM from the SVM^{Light} software package as the classifier with the RBF kernel function. To simulate the masquerade attack detection, a 10 minute sliding window was used throughout the session such that features for the user would be calculated every minute of use. With this method Garg, et al. [22] were able to achieve a 96.15% detection rate.

Before discussing the work done by Imsand, et al. it is important to explain the concept of GUI messages. In the Windows operating system, the kernel produces messages that are sent to applications informing them of user actions that may need responding to. These messages contain information on how the user is interacting with the system, and they can be intercepted and analyzed using hooks [29]. Capturing these messages provides an additional biometric measure that attempts to expand the complete user interaction picture over simply mouse and keyboard dynamics. For example if the user is scrolling a web page it is possible to differentiate between the use of the mouse wheel, paging with the keyboard, clicking the GUI down arrow, or using a click and drag method by monitoring the operating system messages.

The most thorough GUI usage analysis has been done by Imsand, et al. [9, 15–17]. Their work focuses on differentiating between what the users are doing and how they are doing it by monitoring GUI messages. To do this each of the participants used in his research were given a list of tasks to perform. These included word processing, web browsing, searching, and file/folder manipulation within the Windows XP operating system. By doing this it allowed Imsand, et al. to take the actions of the user out of

Table 2.3: Previous work GUI usage features.

Article	Features Calculated	Classifier
Pusara, et al. [7]	The mean, std dev, third moment of distance, angle and speed between a pair of endpoints The mean, std dev, and third moment for X and Y coordinates A count of the number of times each is recorded: - Mouse wheel, clicks outside of the testing area - Single and double clicks for left and right buttons	C5.0 Decision tree
Garg, et al. [22]	- Average number of right and left mouse clicks - Average distance traveled between mouse events - Average movement speed - Movement direction	SVM
Imsand, et, al. [9]	A count of each type of the following: - User actions (clicks, key presses, etc.) - Control types (Buttons, scroll bars, etc.) - Processes/applications that generate messages	Jaccard index and ANN

the equation and purely focus on their GUI interaction style. A total of 31 participants completed the list of tasks five times with 30 minute breaks in between each session. The data was collected by hooking kernel messages inside of Windows as mentioned above. By capturing these messages Imsand, et al. were able to generate a set of features which profiled the user's actions. This included counts regarding the number of times certain user actions, control types and processes were observed, as seen in Table 2.3. Three of the five data sets were used for experimentation. The first two datasets were considered a *block* and features were calculated over each of those datasets. The final set was then used for testing. This style of data set slicing did not line up with some of the other works that focused on dynamic authentication but it is believed this was done to preserve the integrity of the task based testing method. Imsand, et al. used a pre-built ANN that is part of the Matlab numerical analysis suite but was only able to achieve a successful identification rate of 38.7%. It is also important to note that Imsand et al. [9, 15–17] were able to achieve their best results using a Jaccard index which is designed to determine the differences between two data sets. Using this comparison method a 77.1% identification rate was achieved.

Imsand also performed authentication experiments in which a FAR of 8.66% and FRR of 0.0% was produced using term frequency-inverse document frequency (TF-IDF) analysis. Imsand, et al. [9, 15–17] noted that the ANN did not perform as desired due to overfitting.

2.7 Multi-Modal Biometric Techniques

Several instances of research have fused multiple forms of biometric based authentication methods in order to improve the accuracy of the overall system. Asha, et al. [4] combined fingerprint biometrics with mouse dynamics in order to identify the users enrolled in an e-learning class. Rabuzin, et al. [27] also make the case that combining multiple biometric techniques would be beneficial in creating a more robust authentication method for e-learning platforms. Other fusion combinations include voice and facial recognition; fingerprint, voice, and iris; and iris and retinal features [33].

2.7.1 Fusion Methods.

Fusion of biometric modalities can occur in different ways. This has been explored in the physical realm of biometrics but has yet to be heavily tested for behavioral biometrics. According to Ross, et al. [36], in biometric systems fusion can occur by fusing features together, fusing matching scores together, or a fusion of the decisions made by each individual modality. Fusion of features is the simple concatenation of feature vectors from multiple modalities to be input into the classifier [36]. Matching score fusion is specific to physical biometrics so it will not be discussed in this thesis and decision level fusion uses the results from each individual modalities classifier in order to make a final decision [36].

2.7.2 Fusion of Behavioral Biometrics.

Ahmed, et al. [13] integrated keyboard and mouse dynamics into a single architecture that could act as an intrusion detection system. Twenty two subjects were asked to install a monitoring system on their workstations that collected keystrokes and mouse information. They ran the software for nine weeks. In regards to keystroke dynamics Ahmed calculated both latencies and durations for digraphs and trigraphs [2] and used a neural network for classification. For the mouse movements they calculated a subset of features from [14], which appear in Table 2.2 and used a separate ANN for classification. The results from these two classifiers were then presumably combined together via decision level fusion however the exact fusion technique is not discussed by Ahmed, et al [13]. Doing this for all 22 users Ahmed, et al. [13] were able to achieve a FAR of 0.651% and a FRR of 1.312%.

Moskovitch, et al. [30] generated a framework for using behavioral biometrics in computing for the purpose of protecting identity theft. They focused on both keyboard and mouse dynamics but did not perform an experiment to back up their hypothesis. Jagadeesan, et al. [26] researched the concept of user re-authentication that was first defined by Pusara, et al. [7]. They studied a users mouse-to-keyboard interaction ratio to facilitate user identification. The ratio compared the number of mouse events to the number of keyboard events observed by their monitoring software. They used an analysis engine that consisted of statistical analysis, a feed forward neural network with back propagation, and k-nearest neighbor algorithm to achieve a user identification accuracy of 82.2% across all applications tested. They noted it is important to differentiate between the applications tested since the mouse to keyboard ratio is certainly tied to the application being used.

2.8 Concerns Regarding Biometric Authentication

Many of the discussed techniques demonstrate controlled experimental potential but have been untested in the real world due to concerns resulting from biometric based authentication to include: potential vulnerabilities, scalability of the technique, variability of the user, and the privacy of the user's actions.

Potential vulnerabilities exist in many different authentication systems whether it be bugs in the code or a flaw in the overall system architecture. For a biometric system the amount of time it takes to notice that a malicious user has begun interacting with the computer is vital. A skilled hacker with the right tools at their disposal can achieve their malicious intent extremely quick, on the order of minutes. For a system that authenticates every few hours, or even every 30 minutes, a malicious user could be in and out of the network before the biometric system alerts anyone. Current research has not put a large emphasis on collecting data from an increasing number of modalities in order to collect more user data in a shorter amount of time, thus trying to reduce the amount of time needed to both train and test the biometric system.

Scaling the biometric technique into a real world environment has also been largely ignored. Much of the previous research discussed requires a very large amount of training data (months of interaction, or tens of thousands of user actions) for the system to achieve an acceptable level of accuracy. In a real world environment this is not practical. If a system is installed, the consumer would not be pleased with a month long lead time required to achieve a reference signature for each user. The amount of participants used for experimentation has not helped the scalability argument. Many of the previous works use around 20 individuals where as a system deployed to a real world network could need to monitor hundreds.

Variability of the user from day to day, and also over a long period of time, must be taken into account for a biometric system to be considered robust. This is a legitimate concern and other then mentioning the problem there has been a lack of research into how to combat it. If a user breaks their arm, or injures their hand in someway, their typing and mouse dynamics will not be the same [23]. The variability of a user's biometrics over a month or multiple years has also yet to be determined. Marsters [20] had three users in his study record typing dynamics for 18 months and reported that there was little change seen over this period of time however, this remains to be tested on a large scale or for any other modalities.

When recording a users activity the issue of privacy is necessary to discuss. The most common concern is a user's passwords recorded from their keystroke dynamics. To side step this Marsters [20] recording software did not store the chronological keystrokes. It only kept the timing information for all keys and digraphs pressed so that no individual words the user typed could be discovered even if the stored data fell into the wrong hands. In this thesis these privacy issues are noted but are not of concern as test subjects will not be entering sensitive information in the testing environment.

III. Experimental Design

This study fuses data from three modalities, the keyboard, mouse and Graphical User Interface (GUI) to determine if the fused features generated from the keyboard, mouse and GUI increase the performance of a system designed for dynamic authentication. The following chapter presents the data collection method, collection environment and participant tasks, followed by the features generated from the keyboard, mouse and GUI, and finally the fusion system design.

3.1 Data Collection Software

In order to properly identify an individual via a biometric system, data about that user must be collected. The target operating system for data collection in this experiment is Windows 7 particularly because it is used for the Air Force standard desktop and makes up over 45% of the worldwide market share as of January 2013. In the case of this system, there are several different types of actions which dictate how the collection software discussed below is developed. The first is user actions, such as mouse moves, mouse button actions, or keys pressed. The name of the executing process/application and the system time of when the action occurred is required and also the type of GUI control method that was used during the user's interaction. Two different data collection methods were assessed, virtual machine introspection and a Windows driver.

3.1.1 Virtual Machine Introspection.

Virtual machine introspection involves collecting information from the target virtual machine by installing monitoring software on the host machine. Introspection provides several advantages, with the most important being the users working from inside of the guest machines have no knowledge of the monitoring software and no way of tampering with it. As depicted in Figure 3.1, with the monitoring software running in the Control

Domain (Dom0), all keystrokes and mouse events that occur in the Guest Domains could be recorded. Keystrokes are monitored by modifying an open source keylogger for linux called logkeys. This produces output that contains the key, whether it is a press event or release event and the associated system time. To record mouse events a linux program called XMacro was used and modified. XMacro recorded all mouse events as well as coordinates of the pointer every 50 ms. Finally, a customized program that determined if the virtual machine window was in focus was developed. This ensured that data collection only occurs when the user is inside one of the guest domains. All of the data is stored in a text file.

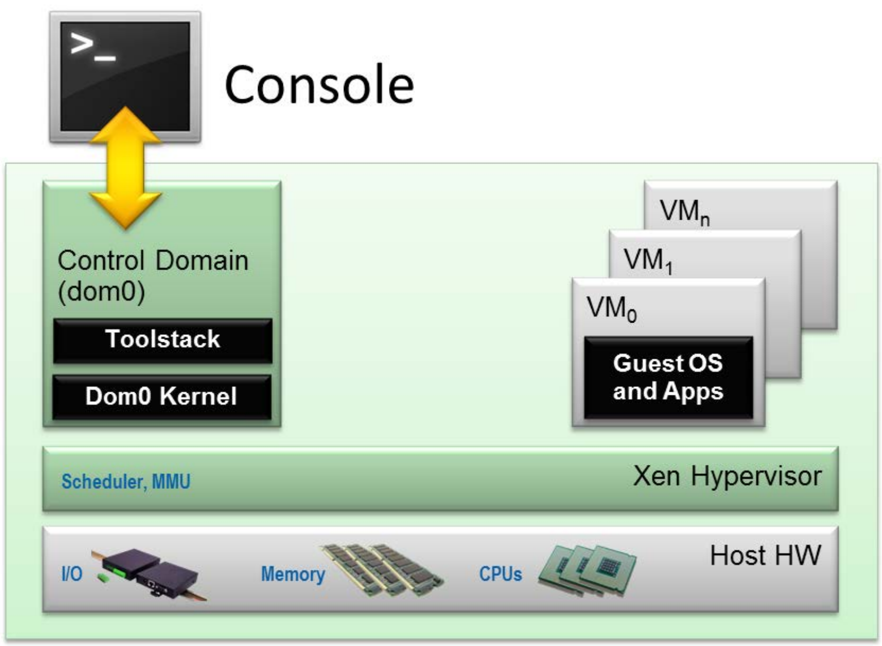

Figure 3.1: The architecture of the Xen hypervisor.

It was determined however, that introspection could not be used to collect the final type of information, GUI control types. This is due to the fact that collecting messages sent internal to the Windows operating system could not be performed fast enough with the current introspection technology. These messages, which contain the type of GUI control method the user employed, in the form of a window class name, could not be captured via introspection.

3.1.2 Windows Driver.

In order to capture all of the necessary messages a Windows driver is developed. The driver works by inserting itself into the Windows hook chain. The Windows operating system maintains a hook chain for each different type of hook that can be installed. When a message is generated that is associated with one of the given hook chains it is passed down the chain so that all applications receive the message appropriately [29]. The hook Application Programming Interface (API) provides the capability to install a hook and monitor system messages for a single application or all applications. In order to install the hook, a call to *SetWindowsHookEx* function is made which places the hook at the top of the specified hook chain. The driver utilizes the WH_GETMESSAGE hook which allows the capture of messages relating to mouse and keyboard input as well as any other messages posted to the message queue [29].

The intercepted messages lack certain necessary information such as the executing process or application for which the message was generated. In order to get the process name the *GetWindowThreadProcessId* and *OpenProcess* functions are used which are provided by the Windows API. The *OpenProcess* function is used to open the process object for reading so that the process name can be extracted.

In order to capture the GUI control types Imsand, et al. [9, 15–17] developed a method that involves monitoring the window class names a user interacts with. In the Windows operating system a window class is used to "define the default behavior of windows belonging to that class" [28]. The Windows API presents a function called *GetClassName* which returns the class name of a given window. Due to this, these classes, and thus the class names are more general and can also change between applications, releases, or even between service pack versions of Windows. For example Imsand, et al. [9, 15–17] who used Windows XP, mentioned class names (text_box, scroll_bar, etc.) that were much more readable than most of the class names discovered in this experiment using Windows 7, as seen in Appendix A. Each application has the ability to register its own window classes with the operating system which can result in a broad range of class names being collected by the software, based on which applications are in use by a given user. A consequence of window class names being general is they do not provide fine grained information such as the name of the control used, therefore, controls like buttons are all counted the same. The full list of class names discovered during testing can be seen in Appendix A.

Table 3.1: Output from the collection tool when recording keystrokes.

Process	Description	Time (ms)	Event
WINWORD.EXE	Microsoft Word Document	1842991521	Keydown m
WINWORD.EXE	Microsoft Word Document	1842991599	Keyup m
WINWORD.EXE	Microsoft Word Document	1842991630	Keydown o
WINWORD.EXE	Microsoft Word Document	1842991677	Keyup o
WINWORD.EXE	Microsoft Word Document	1842991677	Keydown s
WINWORD.EXE	Microsoft Word Document	1842991771	Keydown t
WINWORD.EXE	Microsoft Word Document	1842991786	Keyup s
WINWORD.EXE	Microsoft Word Document	1842991833	Keyup t

Table 3.2: Output from the collection tool when recording mouse events.

Process	Time (ms)	X	Y	Event
chrome.exe	1843274320	1412	1179	MouseMove
chrome.exe	1843274382	1412	1179	MouseMove
chrome.exe	1843274382	1411	1179	MouseMove
chrome.exe	1843274398	1409	1179	MouseMove
chrome.exe	1843274413	1408	1180	MouseMove
chrome.exe	1843274538	1408	1180	LButtonDn
chrome.exe	1843274632	1408	1180	LButtonUp

The software is run by the testing moderator just as any other Microsoft executable, and opens up a blank command shell while the program is executing. Quitting the software is done by closing the command prompt or hitting Ctrl-Alt-F8. During execution all captured messages are written to a text file in Unicode format on the machine where the recording software is running. An example output from the driver can be seen in both Table 3.1 and Table 3.2. The order of the output for each line includes the process name, a brief description if one is available, the time the action occurred (in milliseconds since the system was booted), the X and Y coordinates if the action is mouse related, and the type of event the message was generated for.

The software collects messages as they are sent, so the operating system dictates the resolution at which mouse movement events are recorded. In a typical recorded movement, mouse move are registered about every 20 milliseconds. Key presses, releases, and mouse button clicks are recorded when they are registered by the operating system in both the up and down direction.

3.2 Collection Environment and Task Selection

1. The data collection was performed on a desktop configured with:

    - Windows 7 Service Pack 1
    - Microsoft Office Professional Plus 2010
    - Internet Browsers
        - Internet Explorer 9
        - Firefox 15.0.1
        - Google Chrome 23

The participants used a standard 104 key Windows keyboard, three button mouse with a scroll wheel and had one monitor available with a resolution of 1024x768. Each of the participants were required to perform three separate but similar internet based research tasks. The three tasks asked users to research the pros and cons of installing wind power, solar power and solar water heating at the Air Force Institute of Technology (AFIT) and write a 400-500 word report on each, to include pictures and/or charts to their liking. The exact document given to each participant can be seen in Appendix C. The topic of the scavenger hunt was not essential to the experiment as the main goal was to have the users interact with the machine by doing tasks like searching for text, switching between applications, scrolling documents, choosing the type of applications to use, etc. This type of task, while it is not completely free computer use, is more general than what Imsand used in his GUI study where participants were asked to perform a structured task, but still open enough to resemble a work related task.

In order to ensure that the driver continues to run throughout the testing period a batch script is run. This batch script refreshes the directory where the output files are being stored so that the proctor can ensure the files are continually growing while the participant is working. In order to keep the command window visible a program is used called *Always On Top* which when configured, forces the command window to remain on top of all other windows. The command window is then resized so that it takes up a minimal portion of the screen as seen in Figure 3.2.

3.3 Participant Selection

Thirty one participants came from the general population of AFIT. The majority of the participants were graduate students but there were also instructors, professors and other administrative personnel involved. Since the subjects were all some type of government employee we were able to assume that they had basic computer skills with the Windows operating system to include performing internet searches and the use of a Microsoft Office application for composition. For this reason, no time was allotted for the user to get comfortable with the system.

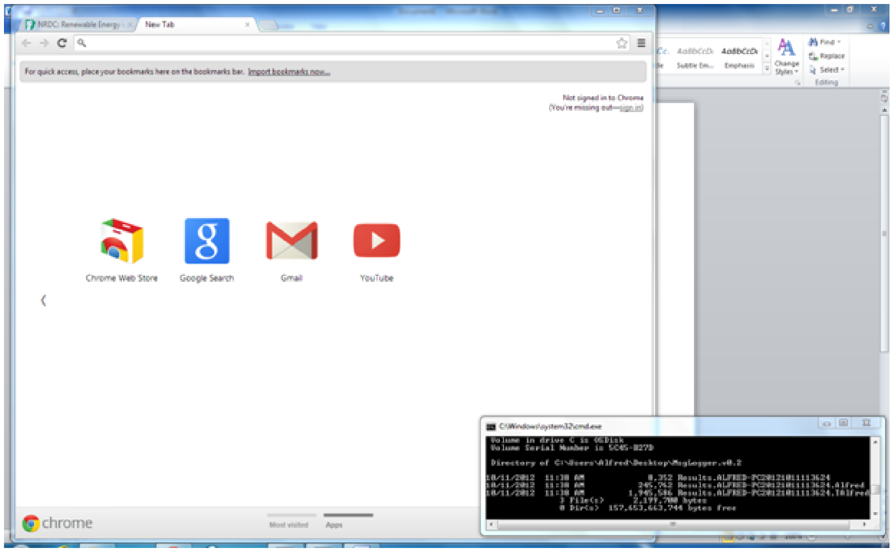

Figure 3.2: The collection environment.

1. Demographics were also taken on the participant population to include:

   - Age range
   - Current profession
   - Highest degree of academic achievement
   - Gender
   - Dominant hand
   - Opinion of computer skills
   - Average daily computer use
   - Source of computer skills
   - Typing rate

Each of these questions are asked in order to asses if certain individuals had any contributing factors from their background that may make them more or less likely to have a higher classification accuracy within the biometric system. By analyzing a users demographics, Imsand [9] was able to discover differences in his systems ability to profile a user. The full survey given to each participant is displayed in Appendix D.

3.4 Feature Generation

After data collection, the raw data is processed to create features for classification. The features calculated are selected from prior works because of their reoccurrence or due to their promising results. As discussed in Section 2.3, previous work has split their collected data up in a variety of ways in order to create a *block* over which a set of features are calculated. In this system the data is chosen to be sliced on a task basis, meaning that a set of features is generated over each of the three tasks a user completes as seen in Figure 3.3, creating three feature vectors per user. This is chosen to ensure that enough data from each modality is collected to allow the features to be consistent and also to not affect the count based GUI features which were previously calculated over a task based feature generation period by Imsand [9, 15–17].

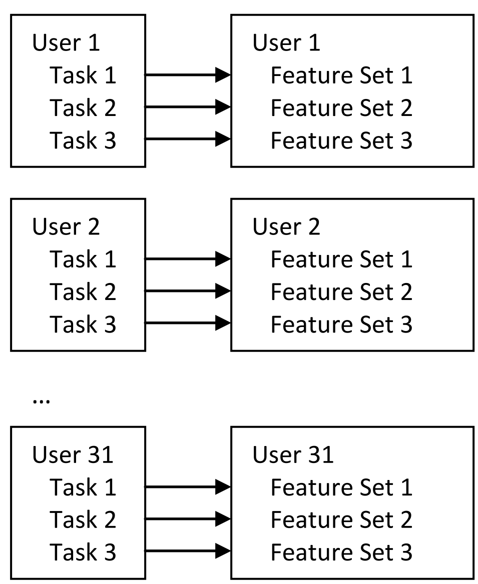

Figure 3.3: Features are generated for each user on a per task basis.

3.4.1 Keystroke Features.

The keystroke features are based off of the work of Marsters [20]. Two different types of features were calculated, durations and latencies, displayed in Table 3.3. Marsters [20] determined that the most consistent keystroke features were seen when the user had pressed a key, or key digraph at least three times in the feature generation period. For this reason the same method is used here and features are not calculated for keys or key combinations pressed less than three times.

3.4.1.1 Keystroke Duration.

Keystroke durations include the mean time that each key is held down which can also be described as the average difference in time between the depress and release of each key. The total time as well as the number of times the key has been pressed is stored for each of the 104 keys. This allows the mean duration of all presses for each key to be calculated.

3.4.1.2 Keystroke Latencies.

Keystroke latencies record the average time it takes for someone to transition between two keys. For example when typing "in" the time between when the user depresses "i" and depresses "n". Using the key down times of every two pairs of adjacently pressed keys an array of latencies can be derived. These are also stored in a similar way as the durations by keeping track of the total latency and number of times the key combination has been witnessed. With a 104 key keyboard this results in 10,816 possible digraph combinations, most of which will never occur. Due to this, any features that never get assigned a value for any user are removed as they do not add value for the classification algorithm.

The user pausing while typing is something that naturally occurs, and it can affect the accuracy of the keystroke latency times. In this type of experiment there are several reasons as to why the user might stop typing, to include if the user begins using the mouse, they get distracted by something around them, or they are just pausing to gather their thoughts. Latencies that have any type of mouse movements in between them are trivial to find and are discarded since it is known that the user has switched away from typing. Hempstalk [23], noticed that users often pause either after pressing the space bar or before pressing the space bar in order to compose their thoughts. Since this has been consistently noticed it is thought that these pauses could add identifiable information to a user's latency times and are therefore not removed on a threshold. If a user pauses due to a distraction in the room or in their mind there is no way to determine this without monitoring them in some other form than our software, such as with a camera, etc. Due to this, these types of pauses are left in the data as they cannot be removed with out using a threshold on latency time.

Table 3.3: Keystroke features.

Feature Type	**Description**
Duration	Average duration each key is held down
Latency	Average latency in transition between two key presses

3.4.2 Mouse Features.

The mouse features were derived from Ahmed, et al. [14], Zheng, et al. [12], and Shen, et al. [19]. The calculation of each feature type is discussed below and listed in Table 3.4. A portion of the features are movement based and require a movement to be defined in order for the feature to be calculated. It was determined that there are two things that can start a movement for the mouse cursor; they are a left button release or a period of silence that goes to movement. If the cursor registers no movement for one second or longer it is deemed to be a period of silence. The value of one second was selected to be a reasonable measure for a period of silence since no previous work defined what interval they used. All events recorded by the driver are in chronological order so movements are discovered by iterating until a movement starter is found. Next the nearest movement ender is located. There are three items that classify as a movement ender. They are a button press, mouse wheel scroll, or mouse silence. The two points representing the start and end of the movement are then saved along with all of the recorded points in between.

3.4.2.1 Average Speed per Movement Direction.

The average speed per movement direction records the users mouse movement speed in eight different directions along the screen which are represented in Figure 3.4. Eight, 45 degree wide sectors were used by Ahmed,et al. [14] so the same is used here. In order to determine the direction of movement, the angle between the coordinates of the movement starter and movement ender is calculated. This is followed by the speed of the movement using the distance formula and the time stamps associated with the beginning and end. The speed values are converted to values that have the units of pixels per second for this calculation and any other speed related values to follow.

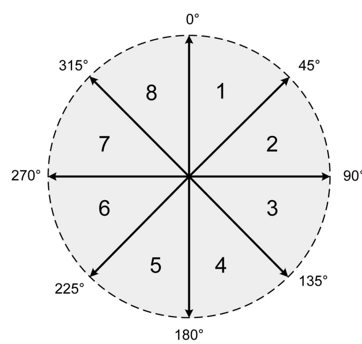

Figure 3.4: Direction sectors of mouse movements [14].

3.4.2.2 Movement Direction Histogram.

The movement direction histogram contains the percentage of movements that the user makes in each of the eight directions.

3.4.2.3 Travel Distance Histogram.

The travel distance histogram contains the percentage of the movements that a user makes in certain distance ranges. All of the distance ranges are measured in pixels. The histogram contains 3 values: short (0-300 pixels), medium (301-600 pixels) and long (601+ pixels). These ranges are from Shen, et al. [19] and due to the resolution of the screen in their testing environment being 1024x768 as well.

3.4.2.4 Distribution of Actions on the Screen.

The distribution of actions made on the screen results in a histogram containing the percentage of movements that end in nine different regions of the screen as seen in Figure 3.5 [19].

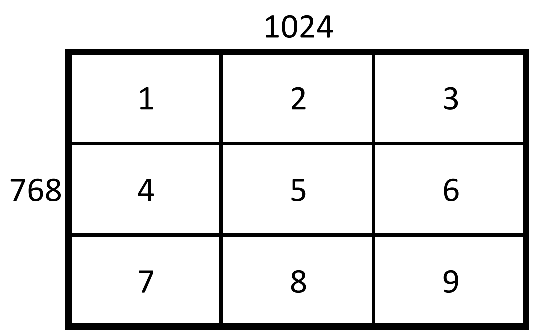

Figure 3.5: The nine screen regions.

3.4.2.5 Single Click Interval Times.

The click interval times are calculated for left and right button single clicks. The single click interval times were calculated by subtracting the time of the down click from the time of the up click, establishing an interval. The average and standard deviation of the intervals for left and right single clicks are calculated and turned into four features.

3.4.2.6 Left Double Click Interval Times.

Double click interval times were calculated by determining the time between all consecutive button down and button ups in the four event sequence. This lead to three different intervals, and the total time is also used creating four intervals which are turned into eight features by calculating the average and standard deviation for each interval.

3.4.2.7 Pause and Click Time.

The pause and click time is the amount of time it takes for the user to click the mouse button after they have stopped moving the cursor. This was shown to be a discriminating feature by Zheng, et al. [12]. The average and standard deviation is taken for both the right and left button, generating four features.

3.4.2.8 Action Histogram.

The action histogram contains the percentage of actions of a given type made by the user. It is made up of five different action types: the number of left, right and double clicks, the number of mouse wheel events, and the number of click and drag actions in where the user holds down the left button while moving the cursor.

3.4.2.9 Extreme Movement Speed Relative to Travel Distance.

The extreme movement speed made by a user in relation to travel distance is similar to the travel distance histogram but instead looks for the largest recorded speed in a given distance range. The same three range lengths are used from the travel distance histogram but with the units of pixels per second.

3.4.2.10 Movement Elapsed Time Histogram.

The time it takes to complete each movement is calculated and stored. Using this stored data, a histogram is created that has information about the number of movements that fall into each time interval. Ahmed, et al. [14] split the histogram up using half second intervals from 0-4 seconds so the same is done here.

3.4.2.11 Average Movement Speed Relative to Travel Distance.

The average movement speed for each travel distance [19] is calculated using previously stored distance and speed calculations about each movement. The same travel distances are used again from the travel distance histogram in order to determine the average speed for short, medium and long movements.

Table 3.4: Mouse modality features.

Feature Description	Calculation Details (# of Features)
Average Speed per Movement Direction [14]	Average velocity in pixels/sec (8)
Movement Direction Histogram [14]	% of movement in each of the 8 directions (8)
Travel Distance Histogram [14]	% of movements occurring in each of 3 distance ranges (3)
Distribution of Actions on Screen [19]	% of actions ending in each of the 9 screen regions (9)
L/R Single Click Interval Times [19]	Avg and St Dev for L/R button click duration (4)
Left Double Click Interval Times [19]	Avg and St Dev for all consecutive presses and releases (8)
Pause and Click Time [12]	Avg and St Dev between when cursor stops and click occurs (4)
Action Histogram [14]	% of time each of the 5 core actions occur (5)
Extreme Movement Speed [19]	Largest recorded velocity (pixels/sec) for each of the 3 distance ranges (3)
Movement Elapsed Time Histogram [14]	Histogram of movements based on elapsed movement time (9)
Average Movement Speed Relative to Travel Distance [14]	Avg movement velocity seen in each of three travel distances (3)

3.4.3 GUI Features.

The features for the GUI usage analysis are calculated by determining the number of times each message occurs as summarized in Table 3.5. This method follows Imsand's process [9], and enumerates differences between the usage styles of different individuals. A counting method is used to translate the text output from the driver into numerical values that the machine learning algorithms can utilize. In order to do this three different classes of items are monitored: user actions, control types and executing processes.

3.4.3.1 User Actions.

This can be any type of user initiated action such as keystroke or mouse event. The counts of each of these separate events are used as the feature values. Unlike for the keystroke features, there is no limit on the number of times the user must make an action for the feature to be counted, all are used.

3.4.3.2 Control Types.

This is represented by a count of each unique type of window class name, which gives a general idea for the GUI buttons and controls that a person uses.

3.4.3.3 Processes Executed.

A count of the number of times each process is seen. This captures what processes or applications the participant is using, as well as a rough estimate on the number of actions that process was used for.

Table 3.5: GUI usage features.

Feature Type	Description
User actions	Count of each type of user action observed (mouse button clicks, key presses, etc.)
Control Types	The number of each type of GUI control message observed (button, scroll bar, ToolbarWindow32)
Processes Executed	The number of times each process name is observed

It is important to note that the entire sample space of the class names and processes is not known before the feature generation occurs, and cannot easily be determined like the entire space of key digraphs. In order to ensure all GUI features are generated, a known values method is used. Each time the feature generator is run it updates a file containing the list of known processes and class names that have been seen. This list is then used to print out the features to the feature file. One disadvantage to listing the GUI features using this known values method is that when a new user's data set is added, if it contains a new known value, all of the feature sets for every user must be updated to reflect the presence of any new features introduced by the new user's data. This is necessary when the number of users who have features calculated is small, however once the number of users grows larger the amount of new known values found is expected to greatly to diminish.

Updating the existing feature sets is done with minimal processing by a program that adds the appropriate number of question marks, annotating missing values, to the end of all previous users feature sets. In the feature file, the GUI related features are printed out last, with the newly found features at the end of the GUI section making them the last features listed for each user. This setup allows for missing values to simply be added at the end of each previous users instance until all contain the proper number of features. Adding missing values works for each user since it is known that no previous users have values for the newly found GUI attributes. It is possible that once a sizable number of window class names have been discovered this process could be stopped and only generate features for the known set class names, ignoring any new class names, however in this experiment it is necessary to ensure that all window class names that get collected are used.

3.5 Fusion System Design

To determine if the fusion of features from all of the modalities (keyboard, mouse and the GUI) provide better results than the individual modalities by themselves, comparison testing of each of the modalities individually, along with two fusion approaches, is performed for multi-class and binary class datasets.

The first fusion approach, seen in Figure 3.7, involves feature fusion, combining all of the features into one file that then has feature selection and classification performed on it to produce results. In Figure 3.8 a decision level fusion technique is presented. Each modality is classified individually, with the results of those classifications being fed into an ensemble classifier that produces the final decision. Both of these experiments use all 31 participants who each provide three data samples for feature generation. All feature selection and machine learning classification is done using the Weka data mining toolkit. Three different classification algorithms are tested; BayesNet, LibSVM and the J48 (C4.5) decision tree. BayesNet was used successfully in keystroke identification by Marsters [20], LibSVM was successfully used by Shen, et al. [19] for classifying mouse dynamics and a variant of the C4.5 decision tree was used by Pusara, et al. [7].

3.5.1 Datasets.

Before discussing the fusion techniques in detail, it is necessary to distinguish between the two datasets that will be tested. Both a multi-class (identification) and binary-class (authentication) dataset are tested for each individual modality and the two methods of feature fusion.

3.5.1.1 Identification.

Identification is a multi-class classification problem. From the data, an ideal classifier distinguishes and identifies the user that generated a given dataset, producing a response of 1-31. Identification classification testing is done with 3-fold cross-validation using two of each users datasets for training and the third for testing.

3.5.1.2 Authentication.

Authentication is a binary classification problem. When using this method a classifier is trained for each individual user. The classifier is given a set of features which represent the known user, and also a sampling of feature sets that represent other users which are not the known user. Remaining feature sets are then fed to the classifier and it is asked to make a decision on whether that data belongs to the known user or not. This type of experiment can lead to two different types of error. Type I error, in which a user who should be authenticated is not, and Type II error in which a user who should not be authenticated is. These errors are represented as the False Rejection Rate (FRR) and False Acceptance Rate (FAR) respectively. The following is broken up into information on how the training set is created for the classifier and what type of data the classifier is tested with.

In order to create the training set a data imbalance needs to be overcome. The data imbalance occurs from distinguishing one user as the known user, and the remaining 30 as unknown users. This leads to only 3 feature vectors for the known user and 90 for the unknown user, which can be seen in Figure 3.6. Due to the small number of training samples that are available per known user, using all of the data for the unknown users creates an imbalance in the training data (2 training sets for the known user and 60 for the unknown users). In order to cut this imbalance down a random sampling of four unknown data sets are used, as seen in Figure 3.6. This reduces the number of unknown to known instances to a 2:1 ratio and improves results.

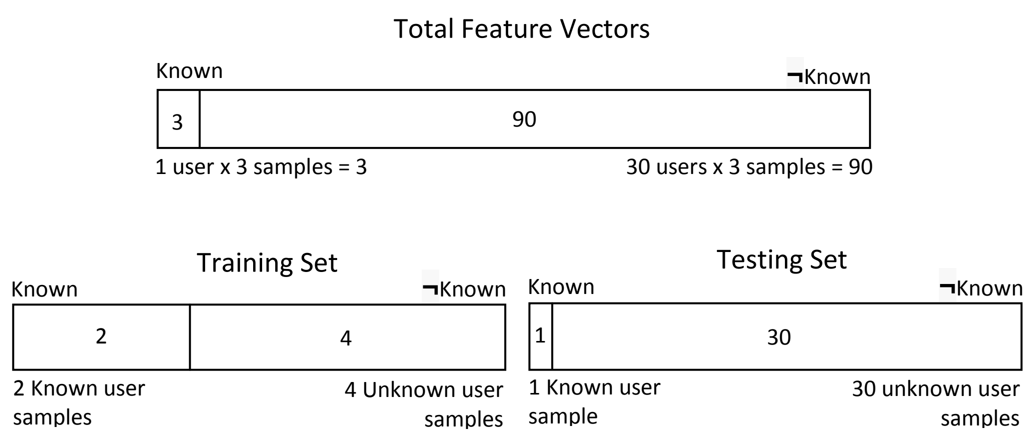

Figure 3.6: The composition of the training and testing dataset for authentication.

The testing data is selected from datasets that have not been used for training. Figure 3.6 shows that the final unused dataset from the known user is added to the test set along with one data set from each of the remaining 30 users, all of which are labeled as unknown users. Using a test set from each user as opposed to just users who have been trained on, allows for many more tests, but more importantly provides a more realistic scenario where the classifier may be encountering testing data for an unknown user that it has not been trained on.

Each of these testing and training cycles are performed three times per user to ensure that all possible combinations of testing and training data are achieved with the three known user samples. Due to the fact that the four unknown training instances, and the 30 unknown testing instances are selected randomly, it is necessary to replicate the dataset multiple times in order to achieve statistical normality. Each of the three known user combinations are replicated 30 times, leading to 90 total training/testing dataset pairs per user. The average of the FAR and FRR is taken over each of the three known user combinations for all 30 replications. This is done for all 31 users. The final FAR and FRR are then calculated by taking the average over all 31 users.

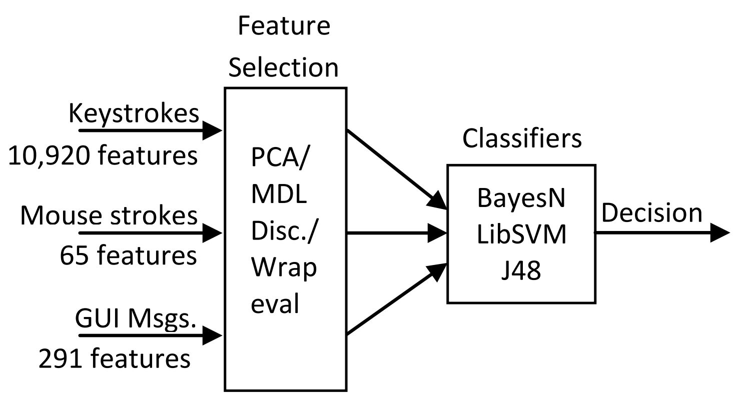

Figure 3.7: Feature level fusion.

3.5.2 *Feature Level Fusion.*

To test the feature fusion method represented by Figure 3.7, all features are combined before any processing occurs. The full dataset is placed into one flat file and feature selection is performed on the data as a whole. Comparison testing of different classifiers is performed using the BayesNet, LibSVM and J48 machine learning algorithms. It should be noted that the type of feature selection used is performed on a per classifier basis. BayesNet requires discretized data so each dataset classified with BayesNet is passed through a supervised discretizing filter based on Fayyad and Irani's Minimum Description Length (MDL) method [21]. LibSVM requires the data to be run through a Principle Components Analysis (PCA) prior to being classified. Finally, since J48 can handle a wide variety of data, the best feature selection method is found through preliminary testing to be a wrapper evaluator using the best first search method. After performing feature selection the final feature count for each classifier is reduced to the number seen in Table 3.6.

Table 3.6: The number of features used by each classifier after feature selection.

Classifier	**Feature Selection**	**Final # of Features**
BayesNet	MDL	330
LibSVM	PCA	80
J48	Wrapper Evaluator	11

3.5.3 Ensemble Based Decision Level Fusion.

In ensemble learning multiple classifiers make decisions on smaller pieces of a larger dataset. These predictions are then combined into a single predictive model which generally will have better performance than any of the individual classifiers alone [32]. In Figure 3.8 the features from each modality are passed through their individual classifier before they are fused together. The type of classifier used for each modality is based on the results from the identification section in Table 4.2. In order to take the output from each of the individual modality classifiers and turn it into usable data for the ensemble classifier, a processing module uses the initial classifiers decisions to generate a set of features which are used by the ensemble classifier.

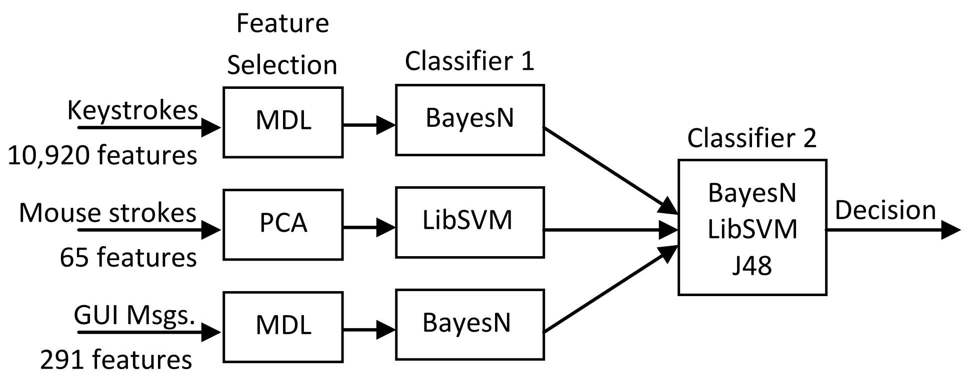

Figure 3.8: Ensemble based decision level fusion.

Three sets of testing and training data are generated for each of the 31 users, and once again they are generated 30 times as discussed in Section 3.5.1.2. This is done for all three modalities individually. Each of the testing and training set pairs, are then run through their respective feature selection methods and classifier. A results file is produced for each training and testing pair with one line of data in the results file per each instance tested from the corresponding test file. Every result contains the class predicted by the classifier for that instance, the classifiers confidence in it's decision represented as a probability, and the actual class of the test instance. An example output is displayed in Table 3.7. The feature file for the final classifier contains three features, consisting of the confidence probability from each of the three individual modality classifiers, and also a class value representing the actual class of the instance. The percentages are expressed with respect to the known user (a probability of 1.0). This means that if the predicted class was an unknown user (a target probability of 0.0), 1 minus the confidence percentage is used, and if the predicted class is the known users, the actual value is used. This is necessary to do in order for the classifier to be able to determine the difference between the data representing the two class values.

Table 3.7: Example output from a individual modality classifier.

Predicted Class	**Probability**	**Actual Class**
1	0.8053219426444527	1
0	0.9860154197012099	0
1	0.6758392920350555	0

It should be noted that it is possible to have a mix of decisions from the initial classifiers as seen in Table 3.8. For example, one modality could predict the data is from the known user, while the other two predict it is from the unknown meaning the initial classifiers have made contradicting decisions. Ideally this allows the final classifier to decide which modality should be alloted more significance in the final predictive model.

Table 3.8: Feature generation example for the ensemble classifier.

Modality	**Predicted Class**	**Probability**	**Actual Class**	**Feature Value**
Key	1	0.9526	1	0.9526
Mouse	1	0.8057	1	0.8057
GUI	0	0.9398	1	0.0602

Since there are 30 unknown user decisions made for every one known user decision, the number of unknown user decisions that are used for creating the ensemble classifiers feature file are reduced. This is done in order to create the best balance of training data between the two class values for the ensemble classifier. With the large number of decisions that occur over all combinations and replications, 10-fold cross-validation is used to assess the performance of the ensemble classifier. The same three classifiers, BayesNet, LibSVM, and J48, are tested as the ensemble classifier seen in Figure 3.8.

IV. Results

This chapter analyzes the classification accuracy of both fusion techniques and determines if either is significantly more effective than the individual modalities on their own. There is also an analysis of the individual modalities performance, comparison of the fusion techniques to previous work, an analysis of how demographics may affect the accuracy of the system, and a alternative way to divide the user generated data.

The 31 test subjects worked on three tasks for an average of 114 minutes. The data collected included 10,446 keystrokes per user with 673 digraphs logged, and 77 of the 104 keys being pressed. An average of 335 mouse movements per session, 23 different processes used, and 147 window class names being registered. Based on this, over nine thousand keyboard features were eliminated due to the fact that no feature values were generated by any of the users. This occurred because of certain keys or two key combinations never being hit by any user throughout the experiment, resulting in the features adding no classification value.

Due to the fact that features were calculated over each task there are only three feature vectors available per user for classification. This small number of feature vectors for each user can cause data imbalance problems between the two class values, as described in Section 3.5.1.2. This small number of feature vectors also required us to use 3-fold cross-validation for identification testing, as opposed to a more standard 10-fold cross-validation approach. Collecting a small amount of user data is not uncommon in this area of research however, mainly due to the use of human subjects. Imsand, et al. [9, 15–17] used the same technique, calculating feature vectors across tasks resulting in the same imbalance, having just three feature vectors per user. Pusara, et al. [7], also used a similar method dividing up their collected data into quarters, using two quarters for training, the third for parameter selection, and the final quarter for testing. Finally, the setup of the authentication dataset is inline with previous work [14, 19, 22] in terms of having to randomly select a subset of the unknown user instances to create a balance between the two class values, known and unknown. While this method does not provide a large amount of data to be used for classification, it has been successfully implemented in previous work without adversely affecting the results. To ensure that system performance did not occur due to chance, statistical testing is also performed throughout this chapter. This is done to confirm that the results are significant, allowing us to be more confident that the small number of user samples did not cause atypical performance of the system.

In Section 4.5 each users data was divided up on a 10 minute time interval for feature generation in order to try and create more samples so that 10-fold cross-validation could be used for testing. However, this method drastically decreased the number of user actions that each feature sample was generated over, hurting overall performance of the system.

Each of the classifiers and feature selection methods were tuned to provide the highest accuracy, with the final parameters shown in Table 4.1. BayesNet was left in its default configuration as provided by Weka. Different estimators and search algorithms were tested but none outperformed the *SimpleEstimator* or the *K2* search algorithm. LibSVM allows for different kernel functions as well as the manipulation of several parameters for each. The sigmoid kernel consistently generated the best results. An experiment was run inside of Weka on the γ parameter and it was deteremined that setting it to 0.01 yielded the highest classification accuracy. The J48 decision tree was tried with several feature selection methods to include ReliefF and a discretization filter; however the wrapper evaluator produced the best results. Parameters were also adjusted to include, using and not using pruning, and adjusting the confidence factor, however none improved results over the Weka defaults.

Table 4.1: Final parameters used for the selected algorithms.

Machine Learning Algorithm	**Final Parameters Selected**
BayesNet	MDL discretization [21] with Weka defaults
LibSVM	Principle component analysis Sigmoid kernel, γ = 0.01
J48	Wrapper evaluator with Weka defaults

4.1 Feature Fusion Results

4.1.1 Identification (Multi-class Dataset).

From the identification results displayed in Table 4.2 it can be seen that the fusion of features, using the method shown in Figure 3.7, performed better than any of the individual modalities on their own. An identification percentage of 97.85% was achieved using BayesNet which outperforms the keystroke, mouse and Graphical User Interface (GUI) modalities when classified on their own. The high fusion percentages validate our hypothesis that by combining features from multiple modalities, classification accuracy can be improved. As can be seen, the keystroke features consistently performed better than the other two modalities which is discussed in Section 4.1.3.

Table 4.2: Identification (multi-class) classification comparison results.

	Identification (3-fold CV) (%)			
	Keyboard	Mouse	GUI	Fusion
BayesNet	94.62 ± 3.72	20.43 ± 3.73	70.96 ± 3.23	97.85 ± 1.86
LibSVM	76.34 ± 1.86	69.89 ± 1.86	45.16 ± 14.78	74.19 ± 11.63
J48	66.67 ± 8.12	30.11 ± 6.72	40.86 ± 1.86	73.11 ± 4.93

Timing data regarding the classifiers is also important to note. As the BayesNet performed much better in accuracy it also produced the quickest total classification time when including the feature selection steps. A table representation of the data can be seen in Appendix F. On average, including discretization, 3-fold cross-validation on the feature fusion identification dataset was performed in 0.93 seconds with MDL discretization [21] taking 1.2 seconds. LibSVM took 6.87 seconds for classification and 32.1 seconds for principle component analysis, and J48 took only 0.56 seconds for classification but over 19 minutes for feature selection using the wrapper evaluator.

4.1.2 Authentication (Binary Class Dataset).

Identifying a user is nice, but being able to authenticate that user is the primary goal behind this system. The results achieved when performing the authentication experiment show similar trends with the multi-class dataset, as seen in Table 4.3. BayesNet outperforms both LibSVM and J48 with a full fusion False Acceptance Rate (FAR) of 3.15% and False Rejection Rate (FRR) of 1.82% when implementing feature fusion. Correcting the imbalance of data when performing the binary class experiment was necessary in order to improve classification performance of the system. As mentioned, the number of unknown user instances included in the training data was four. Figure 4.1 shows that using four instances, or a 2:1 ratio, generates the best results when taking both FAR and FRR into account.

In order to ensure that the fusion results are significantly better than any of the modalities on their own, significance testing is performed over the results of the trials. Significance testing is performed using the Welch two sample t-test. A requirement of the t-test is that the data be normally distributed. In order to determine if the data collected is in fact normal, the Shapiro-Wilk normality test is used. A p-value of 0.78 was achieved, implying that the null hypothesis is rejected (the data is not normally distributed), and accepting the alternative hypothesis that the data is normally distributed. The Welch t-

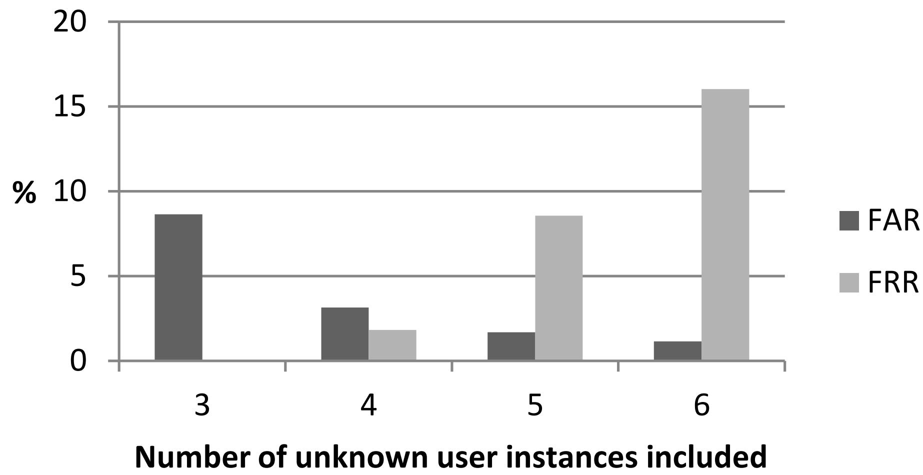

Figure 4.1: Relationship with the number of unknown user instances in the training set and FAR/FRR for feature fusion.

test was selected because it is designed to determine whether a difference in two datasets occurred simply due to chance or not. A standard significance level of 0.05 was selected for the test. The significance results for the fusion technique can be seen in Table 4.4 for FAR and Table 4.5 for FRR.

Table 4.3: Authentication (binary-class) classification comparison results.

Authentication FAR & FRR (%)					
		Keyboard	Mouse	GUI	Fusion
BayesNet	FAR	3.51 ± 0.46	16.22 ± 1.30	18.03 ± 1.12	**3.15 ± 0.55**
	FRR	4.62 ± 1.83	26.70 ± 3.47	20.29 ± 3.13	**1.82 ± 1.13**
LibSVM	FAR	15.52 ± 1.10	0.53 ± 0.28	8.78 ± 1.07	17.13 ± 1.35
	FRR	22.94 ± 3.29	88.24 ± 3.22	45.63 ± 3.84	15.70 ± 3.19
J48	FAR	25.02 ± 1.61	32.03 ± 1.90	27.77 ± 1.73	26.15 ± 1.84
	FRR	23.91 ± 3.48	29.93 ± 4.04	34.23 ± 3.99	26.77 ± 3.85

By looking at each of the p-values in Table 4.4 and Table 4.5 we can see that they are much smaller than the significance level that was set. This means there is convincing evidence that each of the outcomes recorded in Table 4.3 did not occur due to chance. The feature fusion results in Table 4.3 are bolded to represent that they are significantly better than any other results in the table. In Table 4.4 and Table 4.5 all results are recorded with respect to the individual modality data. This means that a confidence interval range of {0.09%, 0.61%} for fusion versus the keystroke modality, means there is 95% confidence that the FAR of the keystrokes will be 0.09% to 0.61% higher than the fusion FAR. Looking at all of the data in Table 4.4 and Table 4.5 it can be seen that the fusion technique is statistically significantly in terms of its effectiveness for authentication when compared to any individual modality by itself. The FAR value for mouse data produces the only negative confidence interval, however this can be discounted because of it's extremely high FRR values.

Table 4.4: Significance of fusion FAR vs individual modalities FAR.

Comparison	**p-Value**	**95% Confidence Interval**
Fusion vs. Key	0.008	{0.09%, 0.61%}
Fusion vs. Mouse	< 0.001	{-2.40%, -2.85%}
Fusion vs. GUI	< 0.001	{14.42%, 15.33%}

Table 4.5: Significance of fusion FRR vs individual modalities FRR.

Comparison	**p-Value**	**95% Confidence Interval**
Fusion vs. Key	< 0.001	{2.01%, 3.58%}
Fusion vs. Mouse	< 0.001	{85.15%, 87.67%}
Fusion vs. GUI	< 0.001	{17.23%, 19.68%}

4.1.3 Individual Modality Performance.

Keystroke features performed the best across all of the classification algorithms mainly because of the large number captured during data collection. Marsters [20] determined that a training block could be calculated effectively with as few as 300 keystrokes. On average our participants generated 3,482 keystrokes per task, exceeding the number needed and improving the classification accuracy of this modality, as it out performed both the mouse and GUI consistently.

The highest identification rate seen for the mouse dataset was 69.89% using LibSVM. This poor performance in comparison with prior work, is attributed to the lack of movements during subject testing. Previous mouse dynamics work [14, 19], required 2,000 mouse actions per training block to achieve their EER of around 1-3 percent. When our users performed the tasks, they generated anywhere from 50-250 movements per task with an average of 335 movements in an entire session. This does not meet the requirements laid out by Ahmed, et al. [14] and Shen, et al. [19] in order to achieve their level of performance and thus resulted in the mouse features under performing.

The point to point mouse features derived by Zheng, et al. [12] were also included in order to gauge their effectiveness. According to Zheng they needed far less testing data than the features derived by Ahmed, et al. [14] and Shen, et al. [19]. Zheng's work achieved an EER of 1.30% using only 25 mouse movements in the test set. After implementing these features, the feature fusion identification results using BayesNet dropped by 4.31%, and Zheng's mouse features on their own were only able to achieve an identification rate of 4.30% using LibSVM. For this reason these features were dropped from the dataset. It is believed that the methods and datasets used for trainging and testing along with the large amount of training data required by Zheng, et al. [12] was the reason for poor performance. Even though the test set only needed 25 mouse movements, their training set still contained 12,500 mouse movements which we were not able to achieve from our data collection. It is also believed that their point to point angle based calculations could vary greatly based on the activity the user is performing at that given time.

The GUI features performed well given the unstructured nature of the task. Imsand achieved an identification rate of 38% with a neural network. Using a BayesNet in this experiment a 71% identification rate was achieved. It is thought that the broader task we selected for the participants accentuated the preferences and tendencies that a user has inside of the GUI. It is also feasible that allowing a user to perform free computer use could further improve these results, however this would need to be tested.

4.1.4 Modality Data Imbalance.

Requiring each of the participants to type 400-500 words is likely what resulted in the imbalance between the number of keystrokes and mouse strokes, as seen in Table 4.6. The participants were encouraged to try and finish each task in 30 minutes in the interest of their own time, but it is possible that this caused them to not fully perform the amount of mouse interaction necessary. The similarity of the tasks also seemed to result in less actions being taken in task three than task one. On average 126 mouse movements were recorded in task one, 118 in task two and just 91 in task three. It is believed that this occurred because of how the participants started to reuse general information pertaining to climate in the area, and general information for introduction and concluding paragraphs.

Table 4.6: Achieved and desired number of actions per training block.

Modality	**Desired Actions**	**Actions Achieved**
Keystrokes	300	3,482
Mouse movements	2,000	111
GUI messages	Not specified	147 window class names 23 processes

A second hypothesis is that once participants found an informative website pertaining to green energy technologies, they were able reuse that website for all three tasks. This would cause them to spend less time researching, resulting in fewer mouse movements. It should also be noted that number of keystrokes recorded declined across the three tasks as well. An average of 3,780 were taken in task one, 3,523 in task two and 3,143 in task three, likely for the same reasons as were just stated. For the GUI messages, there was no threshold set by Imsand because of his structured task based testing however our 31 participants generated messages on 147 unique window class names from 23 different processes, which are displayed in Appendix A and Appendix B respectively.

4.2 Ensemble Based Decision Level Fusion Classification Results

Ensemble Based, Decision Level (EBDL) fusion provides another method for fusing the modalities (Figure 3.8). By combining the modalities together once they have been individually feature selected and classified, it allows for increased accuracy compared to what each of modalities provide on their own, and over feature fusion. The classifiers used for the individual modalities were identified by the performance listed in Table 4.2. BayesNet was selected for both the keystrokes and GUI messages while LibSVM was chosen for the mouse strokes.

The classifier that performed the best as the ensemble classifier was the J48 with bagging (Table 4.7). Bagging, also known as Bootstrap aggregating, generates multiple versions of a classifier, J48 in this case, and uses a plurality voting scheme to make its decision [24]. As with previous authentication tests above, due to the data imbalance per class value, only one unknown test result was randomly selected, along with the one good result to be placed in the ensemble classifiers feature file. As you can see from Figure 4.2, the one-to-one ratio of unknown to known users data performed the best while also outperforming feature fusion with a FAR of 2.65% and a FRR of 1.64%.

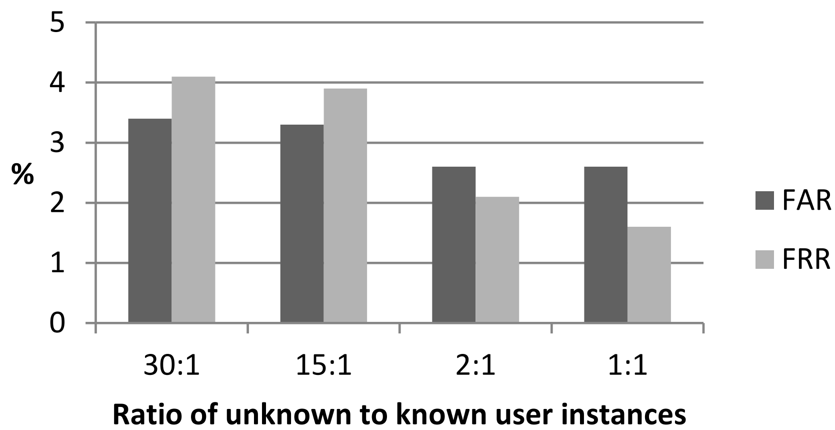

Figure 4.2: Relationship with the number of unknown user instances and FAR/FRR for EBDL fusion.

Significance testing is performed comparing the EBDL fusion method to feature fusion. Once again the Welch t-test was used with the results for EBDL, using the J48 with bagging, and feature fusion using BayesNet since these produced the best accuracy for each technique respectively. This test was performed with a significance level set to 0.05. From Table 4.8 it can be seen that the ensemble based method provides convincing evidence that it is more effective than feature fusion when comparing FAR's, with a p-value of 0.005 and a 95% confidence interval of {0.10%, 0.51%}. When comparing the FRR of the two fusion methods, no significant difference can be seen based on a p-value of .80 and confidence interval that includes zero.

This is statistically significant evidence that EBDL fusion improves accuracy over feature fusion and each of the individual modalities. These values are bolded in Table 4.7 to show that this method is significantly better than all others when considering FAR and FRR. By allowing each of the modalities to be both feature selected and classified in an environment that could produce the best results for that modality, it helped to improve the FAR of classification. For example, when using feature fusion the mouse features were run

Table 4.7: EBDL fusion authentication classification per machine learning algorithm

Ensemble Classifier		Feature Fusion (%)	EBDL Fusion (%)
BayesNet	FAR	3.15 ± 0.55	2.65 ± 0.10
	FRR	1.82 ± 1.13	1.83 ± 0.10
LibSVM	FAR	17.13 ± 1.35	3.42 ± 0.02
	FRR	15.70 ± 3.19	3.41 ± 0.03
J48 with Bagging	FAR	26.15 ± 1.84	**2.65** ± **0.12**
	FRR	26.77 ± 3.85	**1.64** ± **0.11**

through discretization and the BayesNet where they were shown to perform very poorly in Table 4.2. Due to this, it is likely that they did not add any discriminatory value to the feature fusion tests. However, running them through PCA and LibSVM vastly improves their classification accuracy which helped improve the results of EBDL fusion over feature fusion.

Table 4.8: Significance of feature fusion vs. EBDL fusion.

Comparison	p-Value	95% C.I.
False Acceptance Rate (FAR)	0.005	{0.10%, 0.51%}
False Rejection Rate (FRR)	0.80	{-0.35%, 0.46%}

4.3 Comparison with Prior Individual Modality Results

It needs to be noted that the best results presented from previous work on the individual modalities are able to report a better FAR and FRR than appear here. However, to achieve these results months of collection and hundred of hours worth of user interaction needed to be collected for training. In this experiment an average of 76 minutes of interaction was used for classifier training to include about 7,000 keystrokes and 220 mouse actions during that time. In contrast Marsters, et al. [20] collected 18 months of keystroke data to train their system to an Equal Error Rate (EER) of 0.27%, Ahmed, et al. [14] collected over 12 hours of mouse interaction per user to achieve an EER of 2.46% and Shen, et al. [19] recorded data from users for two months resulting in a FAR of 1.86% and FRR of 3.46%. For a system fielded in a real world environment it is not practical to wait this amount of time for the system to become fully trained. By using a fusion of data from multiple modalities a machine learning algorithm requires far less training time to accomplish similar classification results.

The amount of data required for testing is perhaps more important. An experienced malicious user needs only a matter of minutes on a internal computer to impact a network. In a system that is designed to detect and deter an insider threat there needs to be a compromise between the number of actions a user must take to create a testing block, and the accuracy of the system. Some of the previous work require less testing data than our fusion system but this is offset by a very large amount of training data needed as can be seen in Table 4.9. Another benefit to the fusion system is that it is able to capture a malicious user's actions regardless of whether they are using the keyboard or mouse to accomplish their goal. By requiring only 36 minutes of user interaction to achieve an FAR of 2.65% and FRR of 1.64%, fusion from multiple modalities helps to decrease the amount of time required to achieve an acceptable testing set regardless of which modality the user is interacting with.

Due to the classification improvement for these previous systems over time though, it is thought that if more data could have been collected for each user, the classification accuracy would be improved. Table 4.9 shows the number of actions required for both the testing and training set used by previous work along with their best performance classification accuracy. The work done by Imsand, et al. [9, 15–17] does not specify a

Table 4.9: Required number of testing and training actions per previous work.

Previous Work	Training Actions	Testing Actions	Results
Marsters [20]	>85,000 KS	>300 KS, 3 Hrs	EER 0.27%
Ahmed, et al. [14]	10,000 MM	2,000 MM	EER 2.46%
Zheng, et al. [12]	12,500 MM	25 MM	EER 1.30%
Imsand, et al. [9]	N/A	N/A	FAR 8.66% FRR 0.0%
EBDL Fusion	7,000 KS, 220 MM, 72 minutes	3,500 KS, 110 MM, 36 minutes	FAR 2.65% FRR 1.64%

precise number of actions because of their task based testing, so it is not specified. In Table 4.9 KS stands for keystrokes, and MM for mouse movements.

4.4 Demographic Effects on Results

Demographics were taken on each participant in order to assess if there is any correlation between performance of the system and the computer skills of a person, their age, education level, dominant hand, etc. Ideally this information was collected, and the tests performed, to determine if a certain type of user will be predisposed to better or worse results from fusing multi-modal data together. All the collected information per participant can be seen in Appendix E. It was theorized that computer skill and computer use could result as discriminating factors since users who spend more time with a computer may be more likely to act consistently across all three tasks for each of the modalities, keystrokes, mouse dynamics and GUI usage patterns.

1. Tests were performed across the following groups:

   - Opinion of computer skills
   - Daily computer use

- Dominant hand
- Source of computer skills
- Typing speed (< 50, 51-65, 65+) Words per Minute
- Education level
- Age

Comparison testing between each of the possible answers to a demographic question was performed using the results from fusion testing. An Analysis of Variance (ANOVA) test was selected to determine if there are differences between any of the demographic groups and their corresponding identification percentages. A significance level of 0.05 was set for the tests.

Table 4.10: Significance of demographics on classification accuracy.

Demographic Comparison	**p-value**	**Affects Classification**
Opinion of computer skills	0.22	No
Daily computer use	0.93	No
Dominant hand	0.44	No
Source of computer skills	0.89	No
Typing speed	0.97	No
Education level	0.68	No
Age	0.67	No

Based on the significance level that was set, none of the demographics that were recorded provided statistically significant evidence implying they affect the performance of the system which can be seen in Table 4.10. The opinion of computer skills proved to be the most likely to result in a difference, but the p-value was still outside of our significance level. This is a positive result in that based on the demographics collected we can say that the fusion technique does not have a bias to producing better results given the background of the individual. The system worked equally well for all types of users in this experiment.

4.5 Alternate Data Division Results

Generating features using shorter sample time, was done to assess the performance of the system when it has less data for calculating features. In order to produce more feature samples per user, features were generated over a 10 minute time window as shown in Figure 4.3. Each users raw data output from the collection tool was split into 10 minute blocks for feature calculation. Features were generated over each of these 10 minute blocks. This results in an average of nine feature samples per user as opposed to the three on the previous tests. Each of the tests from above are run again on this 10 minute split dataset. This includes the identification and authentication tests for feature fusion and each individual modality, and EBDL fusion. By dividing the data up in to 10 minute intervalsthe samples had an average of 735 keystrokes, 28 mouse movements, and 147 window class names over 23 unique processes applied.

4.5.1 Identification (Multi-Class Dataset).

Identification testing used 10-fold cross-validation to measure performance of three different classifiers, BayesNet, LibSVM, and J48, were tested with feature fusion. Table 4.11 shows that feature fusion improves classification accuracy over each of the individual modalities while using features calculated over 10 minutes of data. It should also be noted that the overall classification accuracy of the system dropped when using less data for calculating the feature sets. The features proved to be less reliable when calculating over this smaller time window than as was shown in previous identification testing (Section 4.1.1), which used features generated over each of the three tasks. However, generating more feature samples allowed for 10-fold cross-validation to be used.

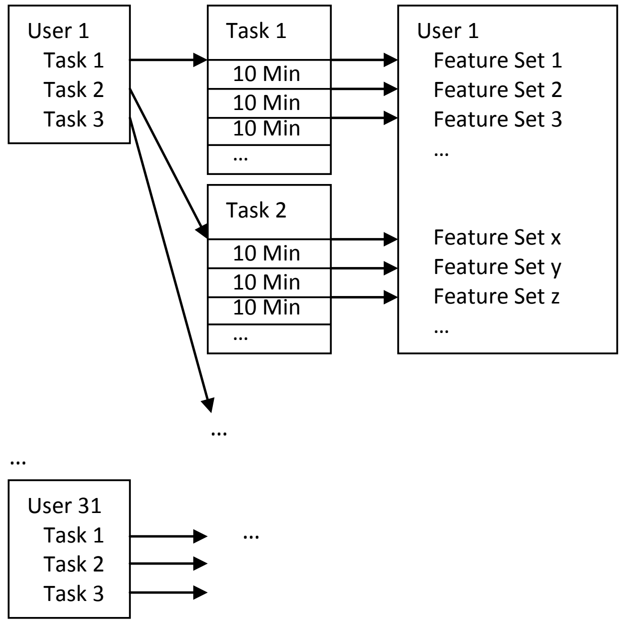

Figure 4.3: Data division and feature generation process for the 10 minute data split.

This creates further confidence that feature fusion will consistently improve identification classification over using an individual modality on its own.

Table 4.11: Identification (multi-class) classification comparison results.

	Identification (10-fold CV) (%)			
	Keyboard	Mouse	GUI	Fusion
BayesNet	91.31 ± 4.01	11.12 ± 4.00	44.81 ± 6.20	95.19 ± 3.37
LibSVM	72.19 ± 6.83	42.85 ± 8.52	38.92 ± 8.59	78.67 ± 6.65
J48	69.65 ± 7.85	25.67 ± 6.73	39.84 ± 6.77	73.32 ± 7.25

4.5.2 Authentication (Binary-Class Dataset).

The authentication dataset was also tested to asses the performance of the fusion system while using 10 minutes of user interaction to generate feature samples. The training and testing sets were as described above, but the number of feature samples in each training and testing set had to be slightly modified in order to adjust for the larger number of feature samples that were available per user. In order to match the experiment discussed in Section 3.5.1.2 (authentication using the task based feature sets), the training and testing sets were created in the following way.

4.5.2.1 Training and Testing Set Creation.

The known users feature samples were divided up into three separate groups with roughly the same number of feature samples in each group. This was done to simulate the three feature samples in the task based feature generation experiment (Section 3.5.1.2). By doing this, two thirds of the known users feature samples could be used for the training set, followed by one third for the testing set. For the following explanation the variable x is defined as follows:

$$x = \left\lfloor \frac{\#ofKnownUserFeatureSamples}{3} \right\rfloor$$

To train the classifier on the unknown user, x feature samples were randomly selected from four unknown users. The testing set then consists of the final third of the known users feature samples, which is of size x, and also one feature sample from all 30 unknown users. Finally, this is replicated over all three possible combinations created by the three groups of known users feature samples, allowing each of the three to be used in the testing set. The three combinations are then replicated 30 times in order to achieve statistical normality due to the random sampling.

By testing this authentication dataset the results displayed in Table 4.12 were achieved. The classification accuracy of all classifiers dropped, similar to the results of the identification dataset, due to the small number of user actions achieved over each feature sample. The results achieved using feature fusion along with the BayesNet for classification are bolded once again to represent that they produced statistically significant improvement over each of the individual modalities.

Table 4.12: Authentication (binary-class) classification comparison results.

Authentication FAR & FRR (%)					
		Keyboard	Mouse	GUI	Fusion
BayesNet	FAR	13.60 ± 0.71	24.83 ± 1.14	26.08 ± 1.09	**12.10 ± 1.13**
	FRR	7.51 ± 1.38	31.48 ± 2.90	28.41 ± 2.59	**5.36 ± 0.93**
LibSVM	FAR	9.73 ± 0.78	16.27 ± 1.14	13.61 ± 1.15	12.80 ± 0.77
	FRR	31.18 ± 2.02	43.18 ± 2.52	50.12 ± 2.89	18.81 ± 2.20
J48	FAR	20.39 ± 1.24	27.76 ± 1.53	21.94 ± 1.48	21.79 ± 1.49
	FRR	18.00 ± 2.25	36.60 ± 3.15	31.89 ± 3.55	17.54 ± 2.43

To perform statistical testing, the data was once again proven to be approximately normal using the Shapiro-Wilk normality test. The Welch t-test was then used in order to asses if the difference between two datasets occurred due to chance or not. The feature fusion dataset was compared to the results achieved for both FAR and FRR. The results from the t-tests can be seen in Table 4.13 for FAR and Table 4.14 for FRR. Feature fusion generates statistically significant improvement over each of the individual modalities in terms of both FAR and FRR.

Table 4.13: Significance of fusion FAR vs individual modalities FAR.

Comparison	**p-Value**	**95% Confidence Interval**
Fusion vs. Key	< 0.001	{1.00%, 1.99%}
Fusion vs. Mouse	< 0.001	{3.57%, 4.75%}
Fusion vs. GUI	< 0.001	{14.12%, 15.27%}

Table 4.14: Significance of fusion FRR vs individual modalities FRR.

Comparison	**p-Value**	**95% Confidence Interval**
Fusion vs. Key	< 0.001	{1.53%, 2.76%}
Fusion vs. Mouse	< 0.001	{36.82%, 38.81%}
Fusion vs. GUI	< 0.001	{22.03%, 24.07%}

4.5.3 Ensemble Based, Decision Level Fusion.

EBDL fusion was also tested using the feature samples calculated over 10 minutes of user interaction in order to asses if it would outperform feature fusion as was seen above in Section 4.2. The individual modality classifiers were selected from the best results seen in Section 4.5.1. BayesNet was used for Keystrokes and GUI features and LibSVM was used for mouse movements. The ensemble classifier was tested using BayesNet, LibSVM, and J48.

From the results displayed in Table 4.15 it can be seen that the J48 with Bagging still produced the highest classification accuracy. Finally Table 4.16 shows that EBDL fusion produces significant improvement over feature fusion in terms of the FAR, however feature fusion produced significant improvement over EBDL fusion in terms of the FRR. Due to this conflicting information, we cannot say for certain that either of the fusion methods performed better than the other using the features calculated over the 10 minute interval. As mentioned above though we can say that both of the fusion methods produced statistically significant improvement over the individual modalities on their own.

4.6 Summary

Collecting user interactions from the keyboard, mouse and GUI, prevents a malicious user from escaping the watchful eye of a system that is able to monitor all three at once. On top of this, GUI usage analysis seeks to emphasize how the user interacts with the

Table 4.15: EBDL fusion authentication classification per machine learning algorithm

Ensemble Classifier		**Feature Fusion (%)**	**EBDL Fusion (%)**
BayesNet	FAR	12.10 ± 1.13	7.28 ± 0.78
	FRR	5.36 ± 0.93	7.78 ± 0.88
LibSVM	FAR	12.80 ± 0.77	9.68 ± 0.84
	FRR	18.81 ± 2.20	7.25 ± 0.76
J48 with Bagging	FAR	21.79 ± 1.49	7.64 ± 0.80
	FRR	17.54 ± 2.43	7.50 ± 0.83

Table 4.16: Significance of feature fusion vs. EBDL fusion.

Comparison	**p-Value**	**95% C.I.**
False Acceptance Rate (FAR)	< 0.001	{3.96%, 4.95%}
False Rejection Rate (FRR)	> 0.999	{-1.69%, -2.59%}

system, such as do they prefer keyboard shortcuts over GUI menus, page up/down versus the scroll bar or scroll wheel, etc. There are thousands of minute differences between how two different users interact with a computer system. Analyzing the entire picture of a users interaction is shown to improve the accuracy and reliability of a behavioral biometric system over using a singular modality. EBDL fusion significantly outperformed each individual modality as well as feature fusion for the task based feature samples, producing a FAR of 2.65% and FRR of 1.64%. These results are in line with previous singular modality work but require less training and testing time to be achieved. By attempting to drop the training and testing time even further, features were generated over a 10 minute interval. This lead to respectable results with the system achieving an EER of 4.81% using BayesNet for Identification, however the classification accuracy of the system was degraded because of the smaller amount of user data that was available for feature generation per sample.

V. Conclusions

Authentication is traditionally based on something you know and/or something you have such as a Common Access Card (CAC) and pin number or a username and password, but biometric authentication relies on something that you *are*. Physical biometrics such as fingerprint scanning and facial recognition have dominated behavioral biometrics which include keystroke dynamics, mouse dynamics, signature recognition, etc. in terms of real world implementation. However, the use of the keystrokes, mouse dynamics and Graphical User Interface (GUI) interaction can be captured from existing hardware throughout a user's session without interrupting them in order to provide active authentication.

5.1 Final Thoughts

There has previously been research into using keystrokes, mouse dynamics, and GUI usage as separate biometric techniques but until now these three modalities have not been combined into a single system. Thirty one participants performed three free computer use research tasks resulting in an average interaction of 114 minutes including 10,446 keystrokes and 335 mouse movements. Using a Windows driver that captures messages sent internal to the Windows 7 operating system, the participants were monitored in three areas, their keystrokes, mouse actions, and GUI usage. Features were then generated on the data gathered for each of the three modalities. Fusing multi-modal data together was performed using feature fusion, where all modalities were feature selected and classified as a whole, and also using Ensemble Based, Decision Level (EBDL) fusion, where each modality was feature selected and classified on its own, with the results being used by a final classifier to make a decision. Both fusion methods were tested, along with the individual modalities on their own in order to determine whether fusion could increase the performance of a biometric system.

The results show that EBDL fusion produced statistically significant improvement over the singular modalities and feature fusion with a False Acceptance Rate (FAR) of 2.65% and a False Rejection Rate (FRR) of 1.64%. While this research has not been able to out perform some of the results produced by the best individual modalities, the amount of training and testing time needed to reach these results is far less than previously seen. This is important to note due to the speed an insider threat would be able to disrupt a network after gaining control of an unlocked computer. The systems ability to collect data from all contact surfaces also makes it more robust than previous work that attempts to combat an insider threat. Regardless of how the malicious user is interacting with the machine, a system that fuses data from multiple modalities will be able to capture it.

The system and experiment discussed present a new approach to combining multiple forms of behavioral biometrics into one architecture. This system fuses data from a users keystrokes, mouse movements and GUI usage together for classification. The fusion of data from multiple modalities requires less time, and user actions, during both training and testing in order to achieve comparable results to previous work.

5.2 Future Work

The end goal of this type of biometric authentication system is to be deployed in a real world environment where users are going about their daily tasks. In order for this to happen several issues need to be addressed in future work. The first is the effectiveness of this system when using free use data. Keystroke and mouse features have both been tested in previous works in a free use environment however it remains to be seen if the GUI features will scale to this type of data. Results are promising though given their improvement from Imsand's [9, 15–17] structured task testing to this experiment. Testing in this free use environment also allows for more data collection per user, this means the system could be tested on more than three feature vectors that were available in this experiment.

Second, is the way the data is divided up into training and testing sets in order to perform active, on-line authentication. Previous work has done this in a variety of ways, none of which has provided a way forward as to which method presents the most effective solution. When considering data coming in from different modalities the data slicing becomes even more difficult. Most published work has calculated features over a block of data based on the number of actions contained in that block. However, a method for doing this across multiple modalities needs to be researched. There are often long periods of time when a user is interacting with only the mouse and not the keyboard or vice versa. This presents a conflict for generating features across all modalities at the same time. When creating this division there is also a balance that must be made between the amount of time it takes to collect one training or testing set and the accuracy of the system. As as shown in Section 4.5 when slicing data over a 10 minute interval the number of actions available for generating features was reduced, which degraded the quality of the feature samples.

The accuracy and amount of training time required for these systems must continue to improve if there is ever a hope for real world deployment. Overwhelming a network administrator or users on a network with false rejections can ruin the usefulness of the system and productivity of the organization. Even a system with a FRR of 0.01% will incorrectly authenticate one user per hour on a network that has 100 users authenticating 10 times per hour. On the contrary though an extremely accurate system that requires on the order of hours of data to authenticate a user is not of use. Skilled malicious users could get their work done and be in another area code before anyone is alerted of what they have done if the testing set requires too much data. As is seen throughout this thesis and previous work there is often a trade off between the number of actions collected and the accuracy of the system.

Appendix A: List of Windows 7 Window Class Names

TaskSwitcherWnd	DDEMLMom	BluetoothNotificationArea-IconWindowClass
Static	ERCAPPIPCRECEIVER	CalcFrame
ATL:000007FEF52DD770	MSTaskListWClass	CiceroUIWndFrame
COMTASKSWINDOWCLASS	Breadcrumb Parent	Progman
MSCTFIME UI	DesktopLogoffPane	DirectUIHWND
msctls_progress32	ANIMATION_TIMER_HWND	AUTHUI.DLL
Desktop top match	Desktop OpenBox Host	#32770
TaskListThumbnailWnd	MSTaskSwWClass	SearchEditBoxWrapperClass
TravelBand	#43	NotifyIconOverflowWindow
DDEMLEvent	Desktop NSCHost	CicMarshalWndClass
TrayClockWClass	FaxMonWinClass	DV2ControlHost
DesktopDestinationList	GDI+ Hook Window Class	CabinetWClass
msseces_class	Button	Groove.Class.BroadcastServices.BroadcastReceiver
OleMainThreadWndClass	tooltips_class32	ReBarWindow32
ShellTabWindowClass	ToolbarWindow32	Desktop More Programs Pane
DesktopSpecialFolders	ExplorerBrowserNavigation	Shell_TrayWnd
MS_WebcheckMonitor	Search Box	CicLoaderWndClass
SHELLDLL_DefView	Desktop User Picture	SystemTray_Main
PNIHiddenWnd	ConsoleWindowClass	ATL:000007FEFB8141F0
LivePreview Address Band Root	Desktop Search Open View	Edit
SysTreeView32	PropertyControlBase	SysListView32
Desktop User Pane	OleDdeWndClass	Media Center SSO
RunDLL	WorkerW	TrayNotifyWnd
Dwm	DesktopProgramsMFU	WindowsUpdate-NotificationWindow
Ghost	ATL:000007FEF6C6D770	ATL:000007FEFB6E41F0

ATL:000007FEF3D452C0	TrayShowDesktopButtonWClass	UniversalSearchBand
NamespaceTreeControl	ATL:000007FEFBC641F0	CtrlNotifySink
ATL:000007FEFC4441F0	PrintCacheListenerWindow	PrintCacheLocalConnection-ListenerHiddenWindow
PrintTray_Notify_WndClass	NManager	LogMeInGui
TASKENGINEWINDOWCLASS	ATL:000007FEFCD841F0	MMDEVAPI
SysShadow	ATL:000007FEFC2F41F0	ScrollBar
ATL:000007FEF2D452C0	#32768	ATL:000007FEFCF441F0
ATL:000007FEFB2441F0	DUIViewWndClassName	SysLink
ATL:000007FEFC4D41F0	ATL:000007FEF5E29D80	Flip3D
SysDragImage	CLIPBRDWNDCLASS	ATL:000007FEF38152C0
ATL:msctls_progress32	ATL:000007FEFBFD41F0	ATL:000007FEF6269D80
ATL:000007FEF2DF52C0	_SearchEditBoxFakeWindow	ATL:000007FEF6274750
IME	Auto-Suggest Dropdown	ComboLBox
ATL:000007FEF5E34750	FloatNotifySink	Notepad
ComboBox	Photos_CommandBar	Photos_NavigationBar
ATL:000007FEF01AA040	ATL:000007FEF01A9E40	ATL:000007FEF01A9400
Photos_ButtonEx	Photos_NavigationPane	Photos_PhotoCanvas
Photo_Lightweight_Viewer	AltTab_KeyHookWnd	ATL:000000013F846DC0
NativeHWNDHost	VANUITooltip	SysTabControl32
ATL:000007FEF5E99D80	ATL:000007FEF61552C0	ATL:000007FEF5EA4750
SyncMgrTrayIconClass	DisplaySwitchUIWnd	ATL:000007FEFB5641F0
ATL:000000013F186DC0	ATL:000007FEF7319D80	ATL:000007FEFB9D41F0
ATL:000007FEF4DA9D80	ATL:000007FEF4DB4750	ATL:000007FEF25E52C0
ATL:000007FEFBEA41F0	ATL:000007FEF56A9D80	

Appendix B: List of Windows 7 Processes

Explorer.exe	calc.exe	WINWORD.EXE
iexplore.exe	ActiveAuthentication.exe	EXCEL.EXE
cmd.exe	POWERPNT.EXE	install_reader11_en_gtba_chra_dy_aih.exe
Eula.exe	setup.exe	chrome.exe
AcroRd32.exe	rundll32.exe	WerFault.exe
ExcelPasswordDemo.exe	Dwm.exe	firefox.exe
notepad.exe	DllHost.exe	sethc.exe
DisplaySwitch.exe	conhost.exe	

Appendix C: Tasks Given to Testing Participants

Task 1

Being "green" can involve several different facets. This could include using an energy source that is sustainable into the future as well as friendly to the environment such as solar, wind or tidal energy. Being "green" can also involve making changes to current architecture of a building or generating new ways to operate in order to consume less energy.

The Air Force Institute of Technology (AFIT) is looking for the best way to become a "green" institution. They need your help determining the return on investment for installing a Wind Turbine behind the facility.

1. The deliverable for this task is a 400-500 word report detailing your findings and recommendation on the best course of action for turning Air Force Institute of Technology (AFIT)into a "green" campus.

    - You should use the internet to find factual information to include in your report. Documentation of your sources does not need to occur but please do not copy and paste information directly from a web page.
    - Factors to take into consideration when making your recommendation
        - Estimated cost of the solution
        - Environmental factors that make a wind turbine efficient
        - Estimated energy savings and/or power generated
        - Life expectancy of the system

The costs and benefits may be best expressed in a table. Also, please include any other information you deem to be necessary. After completing the report, copy it to the given removable hard drive.

Task 2

Being "green" can involve several different facets. This could include using an energy source that is sustainable into the future as well as friendly to the environment such as solar, wind or tidal energy. Being "green" can also involve making changes to current architecture of a building or generating new ways to operate in order to consume less energy.

The Air Force Institute of Technology (AFIT) is looking for the best way to become a "green" institution. They need your help determining the return on investment for installing 50 square meters of solar energy photovoltaic cells on the top of building 642.

1. The deliverable for this task is a 400-500 word report detailing your findings and recommendation on the best course of action for turning AFITinto a "green" campus.

    - You should use the internet to find factual information to include in your report. Documentation of your sources does not need to occur but please do not copy and paste information directly from a web page.
    - Factors to take into consideration when making your recommendation
        - Estimated cost of the solution
        - Environmental factors that make solar cells efficient
        - Estimated energy savings and/or power generated
        - Life expectancy of the system

The costs and benefits may be best expressed in a table. Also, please include any other information you deem to be necessary. After completing the report, copy it to the given removable hard drive.

Task 3

Being "green" can involve several different facets. This could include using an energy source that is sustainable into the future as well as friendly to the environment such as solar, wind or tidal energy. Being "green" can also involve making changes to current architecture of a building or generating new ways to operate in order to consume less energy.

The Air Force Institute of Technology (AFIT) is looking for the best way to become a "green" institution. They need your help determining the return on investment for installing for installing 50 square meters of solar water heating on building 640.

1. The deliverable for this task is a 400-500 word report detailing your findings and recommendation on the best course of action for turning AFITinto a "green" campus.
    - You should use the internet to find factual information to include in your report. Documentation of your sources does not need to occur but please do not copy and paste information directly from a web page.
    - Factors to take into consideration when making your recommendation
        - Estimated cost of the solution
        - Environmental factors that make solar water heating efficient
        - Estimated energy savings and/or power generated
        - Life expectancy of the system

The costs and benefits may be best expressed in a table. Also, please include any other information you deem to be necessary. After completing the report, copy it to the given removable hard drive.

Appendix D: Demographic Survey Given to All Participants

Demographic Survey

Date: _______________

Current Profession: ____

Highest Degree of Academic Achievement: ___________________________________

Gender: M F Abstain

Age: 18-24 25-39 40-65 65+

Dominant Hand: Left Right

How Strong are your computer skills? (Circle one)

Below Average Average Above Average Very Strong

Average daily computer usage in hours: (Circle one)

0-2 2-4 4-6 6-8 8+

What is the source of your computer skills? (Circle all that apply)

Self taught Course/Instruction Computer related degree

What is your average typing speed? Please take the typing test and record your speed here: _______________

Appendix E: Demographic Information for All Participants

1. Legend:

    - ST: Self taught
    - CI: Course/instruction
    - CD: Computer related degree (Computer science, computer engineering, etc.)

ID	Profession	Ed. Level	Gender	Age	Dominate Hand	Opinion of Computer Skills	Daily Computer Use (Hrs)	Source of Computer Skills	Typing Rate
1	Student	M.S.	M	25-39	R	Average	6-8	ST	50
2	Student	M.S.	F	25-39	R	Above avg	6-8	ST,CI	55
3	Student	B.S.	M	25-39	R	Very good	6-8	ST,CI	58
4	IT	B.S.	M	25-39	L	Very good	4-6	CD	40
5	Student	B.S.	M	18-24	R	Average	6-8	ST,CI	67
6	Instructor	M.S.	M	40-65	R	Above avg	6-8	CD	N/A
7	Instructor	M.S.	M	25-39	R	Very good	2-4	CD	N/A
8	Instructor	M.S.	F	40-65	R	Average	4-6	ST,CI	N/A
9	Instructor	B.S.	M	25-39	R	Very good	8+	CD	N/A
10	Instructor	M.S.	M	25-39	R	Below Avg	4-6	ST	N/A
11	Instructor	M.S.	M	25-39	R	Very good	6-8	ST,CI	N/A
12	Student	B.S.	M	25-39	R	Very good	6-8	CD	59
13	Student	B.S.	M	18-24	R	Above avg	2-4	ST,CI	53
14	Engineer	B.S.	F	18-24	R	Average	8+	CI	51
15	Engineer	B.S.	M	25-29	R	Average	2-4	CI	49
16	Student	M.S.	F	25-39	R	Above avg	6-8	ST	68
17	Student	B.S.	M	25-39	L	Very good	6-8	ST	60
18	Professor	PhD	M	25-39	R	Above avg	6-8	ST,CI	90
19	Student	M.S.	M	25-39	L	Above avg	4-6	CI	51

20	Student	B.S.	M	25-39	L	Above avg	4-6	ST,CI	43
21	Student	M.S.	F	25-39	R	Above avg	6-8	ST,CI	54
22	Professor	PhD	M	40-65	R	Very good	2-4	ST,CI	50
23	Student	B.S.	M	25-39	R	Very good	8+	CD	37
24	Staff	PhD	F	25-39	R	Very good	6-8	CI	33
25	Student	B.S.	F	18-24	R	Average	ST	6-8	58
26	Student	M.S.	M	25-39	R	Very good	8+	ST	79
27	Student	B.S.	M	25-39	R	Very good	8+	CD	74
28	Student	B.S.	M	25-39	R	Above avg	6-8	ST,CI	58
29	Student	M.S.	M	25-39	R	Average	4-6	ST,CI	46
30	Student	M.S.	M	25-39	L	Average	2-4	ST	28
31	Student	B.S.	M	18-24	R	Above avg	2-4	ST	44

Appendix F: Multi-class Classification Timing Information Using the Fusion Dataset

Algorithm	Feature Selection Time (sec)	Classification Time (sec)	Total (sec)
BayesNet	1.2	0.93	2.13
LibSVM	32.1	6.87	38.97
J48	1143	0.56	1143.56

Appendix G: Institutional Review Board Approval

DEPARTMENT OF THE AIR FORCE
AIR FORCE RESEARCH LABORATORY
WRIGHT-PATTERSON AIR FORCE BASE OHIO 45433

13 AUG 2012

MEMORANDUM FOR AFIT/CTISR (RONALD F. TUTTLE, PhD)

FROM: 711 HPW/IR (AFRL IRB)

SUBJECT: IRB approval for the use of human volunteers in research

1. Protocol title: Video to Information, Cross-Modal Analysis of Planning Meetings
2. Protocol number: FWR20040019H
3. Protocol version: 10.00
4. Risk: Minimal
5. Approval date: 13 August 2012
6. Expiration date: 12 August 2013
7. Scheduled renewal date: 12 July 2013
8. Type of review: Continuing/Amendment – Expedited
9. Assurance Number and Expiration Date:
   - DoD Assurance #F50301: 29 November 2014
10. CITI Training: Completed
11. The above protocol has been reviewed and approved by the AFRL IRB via expedited review procedures. All requirements, as set by the IRB and its legal counsel, have been fully complied with. The study is about Video Analysis and Content Extraction (VACE) which involves capturing both verbal and visual content fused with the VICON motion tracking system. The data gathered will ultimately yield multi-modal cues to identify fundamental communication events. **Progress:** No subjects have been processed this year, but 105 have been processed overall (with an expected 250 total and potential maximum of 360 subjects). During this time the data collection system has been upgraded to include both active and passive digital cameras and devices that operate in the visible and infrared spectrums. There have been no adverse events. There have been no publications. **Amendments:** Amy Magnus and Christoph Borel-Donohue have been added as Associate Investigators, while Anum Barki, 1Lt Alanna Keith, and 2Lt Kyle Bailey have been added as *engaged* Research Assistants. The Research Monitor position has been deleted as this is a minimal risk protocol (per DoDI 3216.02, a Research Monitor is not required). Verbiage amendments have been submitted to better clarify the

ongoing research. Recruiting email and poster have been updated. This protocol therefore meets the criteria for expedited review in accordance with 32 CFR 219.110 (b)(1) and U.S. Department of Health and Human Services category (7): Research on individual or group characteristics or behavior (including, but not limited to, research on perception, cognition, motivation, identity, language, communication, cultural beliefs or practices, and social behavior) or research employing survey, interview, oral history, focus group, program evaluation, human factors evaluation, or quality assurance methodologies.

12. HIPAA authorization is not required, since no HIPAA protected information will be recorded in the execution of this protocol.

13. FDA regulations do not apply since no drugs, supplements, or unapproved medical devices will be used in this research.

14. This approval applies to human use research (as defined in 32 CFR 219 and AFI 40-402) portions of this project only. Attitude and opinion surveys associated with this research must be conducted IAW AFI 38-501, AF Survey Program. If the study is being conducted under an IDE or IND, a copy of the FDA IDE or IND approval letter must be submitted by the Principal Investigator to the IRB.

15. Any serious adverse event or issues resulting from this study should be reported immediately to the IRB. Amendments to protocols and/or revisions to informed consent documents must have IRB approval prior to implementation. Please retain both hard copy and electronic copy of the final approved protocol and informed consent document.

16. All inquiries and correspondence concerning this protocol should include the protocol number and name of the primary investigator. Please ensure the timely submission of all required progress and final reports and use the templates provided on the AFRL IRB web site http://www.wpafb.af.mil/library/factsheets/factsheet.asp?id=7496 .

17. For questions or concerns, please contact the IRB administrator, Lt Thomas McNitt at thomas.mcnitt@wpafb.af.mil or (937) 904-8100. All inquiries and correspondence concerning this protocol should include the protocol number and name of the primary investigator.

WILLIAM P. BUTLER, Col, USAF, MC, CFS
Director, AFRL IRB

cc:
AFMSA/SGE-C

1st Indorsement to AFIT/CTISR (RONALD F. TUTTLE, PhD) Memo, 13 August 2012, Continuing Review/Amendment expedited approval FWR20040019H

MEMORANDUM FOR 711 HPW/IR (KIM LONDON)

I have reviewed the hardcopy and electronic records and found them to be complete and accurate.

FOR THOMAS C. McNITT, 2 Lt, USAF
Lead Administrator, AFRL IRB

2nd Indorsement to AFIT/CTISR (RONALD F. TUTTLE, PhD) Memo, 13 August 2012, Continuing Review/Amendment expedited approval FWR20040019H

MEMORANDUM FOR AFMSA/SGE-C

This protocol has been reviewed and approved by the AFRL IRB. I concur with the recommendation of the IRB and approve this research.

15 AUG 2012

TIMOTHY J. SAKULICH
Acting Vice Director
711th Human Performance Wing

Bibliography

[1] A., Monrose F. Rubin. "Authentication via Keystroke Dynamics". *ACM Conference on Computer and Communications Security*, 48–56. 1997.

[2] A., Monrose F. Rubin. "Keystroke Dynamics as a Biometric for Authentication". *Future Generation Computer Systems*, 16:351–359, 2000.

[3] Bishop, Matt. *Computer Security: Art and Science*. Addison Wesley, 2003.

[4] C., Asha S. Chellappan. "Authentication of E-Learners Using Multimodal Biometric Technology". *International Symposium on Biometrics and Security Technologies*, 1–6. 2008.

[5] C., Bergadano F. Gunetti D. Picardi. "User Authentication through Keystroke Dynamics". *ACM Transactions on Information and System Security*, 5:367–397, 2002.

[6] C., Gunetti D. Picardi. "Keystroke Analysis of Free Text". *ACM Transactions on Information and System Security*, 8:312–347, 2005.

[7] C., Pusara M. Brodley. "User Re-Authentication via Mouse Movements". *ACM Workshop on Visualization and Data Mining for Computer Security*, 1–8. 2004.

[8] E., Coull S. Branch J. Szymanski B. Breimer. "Intrusion Detection: A Bioinformatics Approach". *IEEE Computer Security Applications Conference*, 1–10. 2003.

[9] E., Imsand. *Applications of GUI Usage Analysis*. Ph.D. thesis, Auburn University, 2008.

[10] G., Joyce R. Gupta. "Identity Authentication Based on Keystroke Latencies". *Communications of the ACM*, 33:168–176, 1990.

[11] Gamboa, Fred A., H. "A Behavioural Biometric System Based on Human Computer Interaction". *Proceedings of SPIE, Biometric Technology for Human Identification*, 5404:381–392, 2004.

[12] H., Zheng N. Paloski A. Wang. "An Efficient User Verification System via Mouse Movements". *ACM Conference on Computer and Communications Security*, 1–12. 2011.

[13] I., Ahmed A. Traore. "Anomaly Intrusion Detection based on Biometrics". *IEEE Workshop on Information Assurance*, 1–7. 2005.

[14] I., Ahmed A. Traore. "A New Biometric Technology Based on Mouse Dynamics". *IEEE Transactions on Dependable and Secure Computing*, 4:165–179, 2007.

[15] J., Imsand E. Garrett D. Hamilton. "User Identification Using GUI Manipulation Patterns and Artificial Neural Networks". *IEEE Symposium on Computational Intelligence in Cyber Security*, 1–6. 2009.

[16] J., Imsand E. Hamilton. "GUI Usage Analysis for Masquerade Detection". *IEEE Workshop on Information Assurance*, 1–7. 2007.

[17] J., Imsand E. Hamilton. "Masquerade Detection through GUIID". *IEEE GLOBECOM*, 1–5. 2008.

[18] J., Shen C. Cai Z. Guan X. Sha H. Du. "Feature Analysis of Mouse Dynamics in Identity Authentication and Monitoring". *IEEE ICC Proceedings*, 1–5. 2009.

[19] J., Shen C. Guan X. Cai. "A Hypo-Optimum Feature Selection Strategy for Mouse Dynamics in Continuous Identity Authentication and Monitoring". *IEEE International Conference on Information Theory and Information Security*, 349–353. 2010.

[20] J.D., Marsters. *Keystroke Dynamics as a Biometric*. Ph.D. thesis, University of Southampton, 2009.

[21] K., Fayyad U. Irani. "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning". *Thirteenth International Joint Conference on Artificial Intelligence*, 1–6. 1993.

[22] K., Garg A. Rahalkar R. Upadhyaya S. Kwiat. "Profiling Users in GUI Based Systems for Masquerade Detection". *IEEE Workshop on Information Assurance*, 1–7. 2006.

[23] K., Hempstalk. *Continuous Typist Verification using Machine Learning*. Ph.D. thesis, The University of Waikato, 2009.

[24] L., Breiman. "Bagging Predictors". *Machine Learning*, 24:123–140, 1996.

[25] M., Bhattacharyya D. Ranjan R. Alisherov F. Choi. "Biometric Authentic: A Review". *International Journal of Service, Science and Technology*, 2:13–28, 2009.

[26] M., Jagadeesan H. Hsiao. "A Novel Approach to Design of User Re-Authentication Systems". *IEEE Conference on Biometrics: Theory, Applications and Systems*, 1–6. 2009.

[27] M., Rabuzin K Baca M. Sajko. "E-Learning: Biometrics as a Security Factor". *International Multi-Conference on Computing in the Global Information Technology*, 1–6. 2006.

[28] Microsoft. "About Window Classes". Microsoft Developer Netowrk, October 2012.

[29] Microsoft. "Hooks Overview". Microsoft Developer Network, September 2012.

[30] Mosokovitch R. Feher C. Messerman A. Kirschnick N. Mustafic T. Camtepe A. Lohlein B. Heister U. Moller S. Rokach L. Elovic, Y. “Identity Theft, Computers and Behavioral Biometrics”. *IEEE Conference on Intelligence and Security Informatics*, 155–160. 2009.

[31] N., Gaines R. Lisowski W. Press S. Shapiro. *Authentication by Keystroke Timing: Some Preliminary Results*. Technical report, Rand Corporation, 1980.

[32] Opitz, R, D. Maclin. “Popular Ensemble Methods: An Empirical Study”. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[33] P., Singhal R. Singh N. Jain. “Towards an Integrated Biometric Technique”. *International Journal of Computer Application*, 42:20–23, 2012.

[34] R., Matyas V. Zdenek. “Toward Reliable User Authentication through Biometrics”. *IEEE Security and Privacy*, May/June:45–49, 2003.

[35] R., Shen C. Cai Z. Guan X. Du Y. Maxion. “User Authentication Through Mouse Dynamics”. *IEEE Transactions on Information Forensics and Security*, 8:16–30, 2013.

[36] Ross, A, A Jain. “Information Fusion in Biometrics”. *Pattern Recognition Letters*, 24:2115–2125, 2003.

[37] S.J., Brown M. Rogers. “User Identification via Keystroke Characteristics of Typed Names using Neural Networks”. *International Journal of Man-Machine Studies*, 39:999–1014, 1993.

[38] T., Maxion R. Townsend. “Masquerade Detection Using Truncated Command Lines”. *IEEE International Conference on Dependable Systems and Networks*, 1–10. 2002.

[39] X., Shen C. Cai Z. Guan. “Continuous Authentication for Mouse Dynamics: A Pattern-Growth Approach”. *IEEE International Conference on Dependable Systems and Networks*, 1–12. 2012.

[40] Y., Schonlau M. DuMouchel W. Ju W-H. Karr A. Theus M. Vardi. “Computer Intrusion: Detecting Masquerades”. *Statistical Science*, 16:1–16, 2001.

Vita

Kyle Bailey is from Austin, Texas and graduated from the Unites States Air Force Academy in 2011.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
21–03–2013	Master's Thesis	Oct 2011–Mar 2013

4. TITLE AND SUBTITLE

Computer Based Behavioral Biometric Authentication
via Multi-Modal Fusion

5a. CONTRACT NUMBER
AFOSR/LRIR 2010-097

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Bailey, Kyle O., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-13-M-04

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr. Tristan Nguyen
Air Force Office of Scientific Research
875 N. Randolph, Ste.325
Arlington Virginia, 22203

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR/RTC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Biometric computer authentication has an advantage over password and access card authentication in that it is based on something you *are*, which is not easily copied or stolen. One way of performing biometric computer authentication is to use behavioral tendencies associated with how a user interacts with the computer. However, behavioral biometric authentication accuracy rates are much larger then more traditional authentication methods. This thesis presents a behavioral biometric system that fuses user data from keyboard, mouse, and Graphical User Interface (GUI) interactions. Combining the modalities results in a more accurate authentication decision based on a broader view of the user's computer activity while requiring less user interaction to train the system than previous work. Testing over 30 users, shows that fusion techniques significantly improve behavioral biometric authentication accuracy over single modalities on their own. Two fusion techniques are presented, feature fusion and decision level fusion. Using an ensemble based classification method the decision level fusion technique improves the FAR by 0.86% and FRR by 2.98% over the best individual modality.

15. SUBJECT TERMS

Biometrics, Computer Security, Insider Threat, Active Authentication

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER *(include area code)*
U	U	U	UU	101	Dr. Gilbert L. Peterson (ENG) (937) 255 -6565 x4281 gilbert.peterson@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18